AFRL-RQ-WP-TR-2016-0140V1

AIRFRAME DIGITAL TWIN (ADT)

Delivery Order 0001: Scalable, Accurate, Flexible, Efficient, Robust, Prognostic and Probabilistic Individual Aircraft Tracking (SAFER-P^{2}IAT)

Volume 1

Liping Wang, Isaac Asher, K evin Ryan, and Genghis Khan

The General Electric Company
GE Global Research

Dale Ball

Lockheed Martin Corporation

SEPTEMBER 2016
Final Report

AIR FORCE RESEARCH LABORATORY
AEROSPACE SYSTEMS DIRECTORATE
WRIGHT-PATTERSON AIR FORCE BASE, OH 45433-7542
AIR FORCE MATERIEL COMMAND
UNITED STATES AIR FORCE

NOTICE AND SIGNATURE PAGE

REPORT DOCUMENTATION PAGE	*Form Approved* *OMB No. 0704-0188*

1. REPORT DATE ***(DD-MM-YY)*** September 2016	**2. REPORT TYPE** Final	**3. DATES COVERED** ***(From - To)*** 06 November 2013 – 01 August 2016

4. TITLE AND SUBTITLE AIRFRAME DIGITAL TWIN (ADT) Delivery Order 0001: Scalable, Accurate, Flexible, Efficient, Robust, Prognostic and Probabilistic Individual Aircraft Tracking (SAFER-P^{2}IAT) Volume 1	**5a. CONTRACT NUMBER** FA8650-14-D-2443-0001
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER 62201F
6. AUTHOR(S) Liping Wang, Isaac Asher, Kevin Ryan, and Genghis Khan (GE Global Research) Dale Ball (Lockheed Martin Corporation)	**5d. PROJECT NUMBER** 2401
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER Q0Q8
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) The General Electric Company, GE Global Research, 1 Research Circle K1-4B5A, Niskayuna, NY 12309 — Lockheed Martin Corporation	**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Air Force Research Laboratory, Aerospace Systems Directorate, Wright-Patterson Air Force Base, OH 45433-7542, Air Force Materiel Command, United States Air Force	**10. SPONSORING/MONITORING AGENCY ACRONYM(S)** AFRL/RQVS
	11. SPONSORING/MONITORING AGENCY REPORT NUMBER(S) AFRL-RQ-WP-TR-2016-0140V1

12. DISTRIBUTION/AVAILABILITY STATEMENT
DISTRIBUTION STATEMENT A: Approved for public release. Distribution is unlimited.

13. SUPPLEMENTARY NOTES
PA Case Number: 88ABW-2018-5329; Clearance Date: 23 October 2018.

© GE Global Research 2016. This work was funded in whole or in part by Department of the Air Force contract FA8650-14-D-2443-0001. The U.S. Government has for itself and others acting on its behalf a paid-up, nonexclusive, irrevocable worldwide license to use, modify, reproduce, release, perform, display, or disclose the work by or on behalf of the U.S. Government.

See also AFRL-RQ-WP-TR-2016-0140V2.

14. ABSTRACT
Improving the accuracy of structural diagnosis and prognosis to make better maintenance decisions is accomplished through more realistic structural analysis of fatigue crack growth, including sources of uncertainty into predictions, and fusing usage and inspection data to update and reduce uncertainty. A set of methods for uncertainty quantification and updating form the basis of the framework. Modularity supports transition to other platforms and reliability problems. Uncertainty in inputs and outputs is described by parametric or non-parametric probability distributions. Criteria for performing inspections can be established based on probabilities of events.

15. SUBJECT TERMS
uncertainty quantification, Bayesian updating, inspection scheduling, probabilistic crack growth

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT:**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON** (Monitor)
a. REPORT Unclassified	**b. ABSTRACT** Unclassified	**c. THIS PAGE** Unclassified	SAR	135	Scott R. Wacker **19b. TELEPHONE NUMBER** (Include Area Code) N/A

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std. Z39-18

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

This report is based upon work supported by ARFL under Contract FA8650-14-D-2443, under the direction of Dr. Pam Kobryn, Dr. Eric Tuegel and Mr. Doug Henderson.

The authors would like to thank the following for their contributions to the Airframe Digital Twin project that helped make it a success. From GE Global Research, the original team which formed the basis of our approach was Felipe Viana and Sergio Choze. All the finite element work at GE was performed by Changjie Sun and Sunilkumar Soni. Much of the original code for the DBN was developed by You Ling. Additional help was given by Youngwon Shin.

At Lockheed Martin, Don Dubowski shared engineering insight and helped run many analyses. Elias Dakwar helped with control point selection and loads development.

At Wyle Aerospace, Jochen Hoffman was our knowledgeable NDI expert.

At Unison, Hua Jiang was always helpful and retrieved all blueprint drawings.

At Vanderbilt University, Professor Sankaran Mahadevan provided guidance on methods and early proofs-of-concept. Chenzhao Li interned with GE, worked on a GUI, and developed the sensitivity methods.

At the Air Force Research Lab, the entire digital twin team has been wonderful to work with. They have consistently provided guidance and insight and pushed us to do our best. We remember Ned Lindsley fondly and appreciate his insight into fighter aircraft systems and analyses. His practical and straight forward advice kept the project going forward. Daniel Sparkman provided mathematical insight and encouraged excellent discourse. We also thank Scott Wacker for taking over an important role with seriousness, commitment, and a strong vision.

It is difficult to overstate the impact of Pam Kobryn, Eric Tuegel, and Doug Henderson. Their vision of digital twin inspired the team and much of the engineering world. Their relentless and detailed work has brought the concept of digital twin to life.

1.0 EXECUTIVE SUMMARY

The overall goal of the Airframe Digital Twin program is to improve the accuracy of structural diagnosis and prognosis in order to make better maintenance decisions. This is accomplished by more realistic structural analysis models of fatigue crack growth, including a variety of sources of uncertainty into predictions, and fusing usage and inspection data to update and reduce uncertainty in predictions. The integrated models comprise the Prognostic and Probabilistic Individual Aircraft Tracking (P^2IAT) framework. A set of methods for uncertainty quantification and updating form the basis of the framework. Modularity allows the methods to be easily applied to other platforms and reliability problems. Uncertainty in input parameters and output variables can be described by parametric and non-parametric probability distributions. Criteria for performing inspections can be established based on probabilities of events or known schedules.

The functionality of the SAFER-P^2IAT framework is demonstrated in the Task Order 0001 effort for a single location.

2.0 INTRODUCTION

The first task in the Prognostic and Probabilistic Individual Aircraft Tracking - SAFER-P^2IAT is the development of probabilistic aircraft usage and loads, followed by the probabilistic structural reliability analysis, statistical updating, and decision making analysis. This report documents how several engineering analyses and models were integrated using state-of-the-art statistical methods. In particular, a detailed method for developing probabilistic aircraft usage loads using both flight records and a flight simulator is developed. Supporting analysis specifically tailored to estimating structural reliability are discussed, including finite element and fatigue crack growth models. A methodology for using inspection data consisting of yes/no detections for updating the probabilistic models is developed with the aid of the probability of detection curve. Inspection scheduling is performed by computing probability of failure and estimating the effects of future inspections, including both the expected updates and repairs. The methods and models are integrated into a framework which enables data assimilation, updating, forecasting, and maintenance planning. The algorithms were optimized to reduce both time and memory consumption and utilize a parallel Python implementation.

The methods are demonstrated on a single location on the outer wing of a fighter aircraft. The loading, geometry, material properties, initial flaw size, and inspection data inputs were all estimated as accurately as possible using engineering data for the aircraft. Models are developed to compute external loads, stress intensity factors, and fatigue crack growth rates. Crack growth and reliability computations are performed for assumed flight recorder data as well as forecasting using the statistical usage loads. The results of the SAFER- P^2IAT framework include the estimated crack length distributions, the forecasted inspection schedule, reliability over time, and sensitivity of outputs to input parameter distributions.

3.0 BASELINE SPECTRUM

A key input to any tool estimating reliability is usage data. That is, an accurate estimate of reliability of a particular asset must take into account the usage history of that asset. In the Prognostic and Probabilistic Individual Aircraft Tracking – (SAFER- P^2IAT) framework, a probabilistic usage model is developed. The probabilistic load spectrum, or baseline usage data, will be used as a baseline usage assumption for the asset to be modelled in P^2IAT. It will be used to fill in missing flights and to forecast flights into the future. The spectrum can be adapted to actual usage by tagging flights with a mission ID and fitting a mission mix to a given asset. In this way, filling missing data and forecasting can be tailored to how a particular asset is operated.

The first pass at designing the probabilistic load spectrum tackles a key step with a reduced scope. The key step is combining data from a six degree-of-freedom (6DOF) flight simulator (Stick-to-Stress V1.0) with actual flight data. We assumed that the simulator calculated realistic maneuver loads. The flight recorder data provided by the Air Force had been processed to compute approximate loads. The processed loads will not be available during real operation of P^2IAT – rather, the P^2IAT framework will be required to use standard flight recorder data as input, which normally does not include loads. However, other flight parameters which are measured by the flight recorder should be representative of the aircraft usage. By connecting the flight simulator and the flight recorder data, we recovered a realistic load spectrum. At the same time, we fit statistical models to the usage to construct the probabilistic load spectrum.

Next consider the two inputs to the analysis, the flight recorder data and flight simulator results. The probabilistic load spectrum is built using all flight records provided by the Air Force. The second input is the 6DOF flight simulator results from 110 different maneuver settings. The inputs used to generate these maneuvers are listed in Appendix A.

The set of maneuvers was designed to have sufficient coverage of the usage envelope. The probabilistic load spectrum is constructed by extracting load content flight simulator runs and maneuver sequence information from the flight records. Two statistical models were fit, one to the maneuver sequencing and the second to steady flight loads. Finally, draws from the probabilistic load spectrum, consisting of simulated load histories, were compared to the actual flight data in terms of exceedance curves and crack growth. The goal is to produce a probabilistic load spectrum which is reasonably similar to actual aircraft usages in terms of mean and variation.

The overall process for designing the probabilistic load spectrum is shown in Figure 1. First, the flight record is split into steady flight portions and maneuver portions by comparing the flight recorder data in the flight record to those in the flight simulator maneuvers. Each time hack in the flight recorder data is identified either with a maneuver or with steady flight. Next, the sequence of maneuvers is used to build a probabilistic maneuver model using a Markov process. The Markov process is then modified for each mission type. The steady flight portions are modeled separately using distributions of load cycles. The maneuver and steady models are then combined via an interspersing method. Finally, each draw from the probabilistic load spectrum results in a load history for a single flight of a particular mission type.

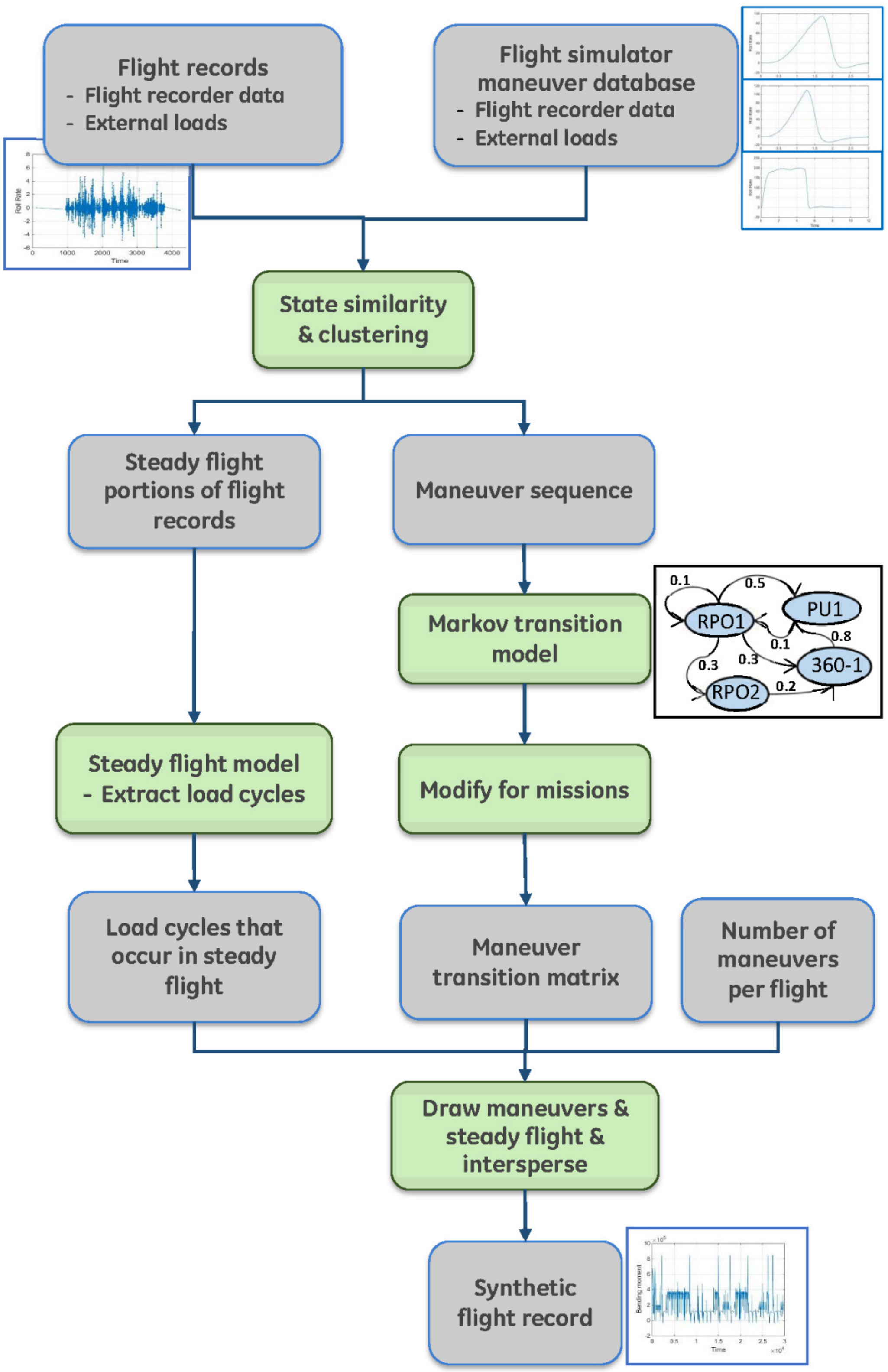


Figure 1. Process for Designing the Probabilistic Load Spectrum.

3.1 Maneuver Sequence

The first step in processing the flight recorder data is to extract the maneuver sequence information. To do this, the aircraft state from the flight record is matched to states in the flight simulator maneuvers. In a sense, this process identifies which maneuvers occurred in the flight record. However, it should be emphasized that the end goal is really to identify the load state of the aircraft, not the maneuver per se. Maneuvers are used because they happen to be a convenient way to group and identify load states, and because the flight simulator runs on a maneuver basis.

The aircraft states used for comparison are chosen from two criteria:

1. States must be measured by flight data acquisition systems (i.e. not loads, which are added in a post-processing step).
2. States must also be inputs or outputs of the flight simulator.

States are also chosen to clearly differentiate and identify maneuvers, since matching will be done according to the states. The chosen states are roll angle, roll/pitch/yaw rates, roll acceleration, pitch acceleration, altitude, Mach number, and NNZCG (vertical acceleration at the center of gravity). It is straightforward to re-run the analysis with a different set of states in the future.

Next, the states are compared using a similarity metric that normalizes and decorrelates them – the Mahalanobis distance [1]. The Mahalanobis distance between a flight record state f and a flight simulator state s is

$$d_{mahal}(f,s) = \sqrt{(f-s)S^{-1}(f-s)} \tag{1}$$

where S is the sample covariance matrix of the states (in this case, just the flight simulator states). The distance between every flight record time hack and every flight simulator time hack (in each maneuver) is computed. A candidate maneuver is identified at each flight record time hack as the maneuver with the smallest $d_{mahal}(f,s)$. This results in a sequence of maneuver IDs at every time hack in the flight record data.

The next step is to group time hacks together into full maneuvers so that transitions between maneuvers can be identified. Each flight simulator maneuver lasts approximately 3 seconds, which comprises about 30 time hacks in the flight recorder data. To group together the time hacks, a clustering algorithm is used. Clustering the time hacks also removes some time hacks which may be misidentified. The clustering method [2] attempts to minimize the distance from each point to its cluster's center. In this case, each cluster consists of time hacks with values of time and maneuver ID. The distance between time hacks with times t_1, t_2 and maneuver IDs m_1, m_2 is calculated as:

$$d_{cluster}((t_1,m_1),(t_2,m_2)) = (t_2 - t_1) + \begin{cases} 1, & m_1 \neq m_2 \\ 0, & m_1 = m_2 \end{cases} \tag{2}$$

This distance metric incorporates the logic that two maneuvers are either the same or different, but there is no concept of a distance between maneuvers as such. In the clustering algorithm, the number of clusters n_c must be fixed a-priori, and the size of each cluster is an output. To get appropriate cluster sizes, the clustering is done with three settings of n_c, and the average cluster size is computed for each. Typically, there is a smooth relationship between n_c and the average cluster size. By fitting a line to this relationship, the value of n_c for which the average cluster size is 3 seconds can be found. The clustering algorithm is re-run the final time with this optimal value of n_c.

An additional filtering step is done to ensure that the same maneuver is not repeated many times, as this is considered unrealistic. Note that the dictionary of maneuvers contains, for example, many roll-pull-outs at various Mach and altitude starting points, so these are considered different maneuvers. The unrealistic case, then, is when the roll-pull-out is started at exactly the same conditions multiple times in a row, whereas a series of roll-pull-outs at different initial conditions is possible. The result of the clustering step is a sequence of clusters for the flight recorder data, each cluster identified with a single maneuver. That is, the result is a sequence of maneuvers.

The third step is to fit a statistical model to the maneuver sequence. The goal of this step is to finally extract the sequence information from the flight recorder data. The statistical model chosen is a Markov process which models transitions over time [3]. This is a general model which defines a probability of transitions, in this case from one maneuver to the next. There are $n_m = 110$ maneuvers, so the model is fully described by a 110-by-110 transition matrix. Row i of the matrix contains the probabilities of transitioning from the i^{th} maneuver to any other maneuver. Fitting the model is simply done by counting the transition in the actual flight data and converting the counts into a probability. For example, suppose the maneuvers are identified as A, B, C and the processed flight data results in the sequence AABCBABCBACAC. The number of transitions from A to A is 1, from A to B is 2, and from A to C is 2. Then, the first row of the transition matrix is [1/5, 2/5, 2/5]. The other rows are constructed in the same way.

To draw a sample from the Markov process, one begins at an arbitrary initial maneuver (possibly sampled from the actual first maneuvers identified in the flight records) and picks the next maneuver according to the probabilities of the appropriate row. The only question left to answer is how many maneuvers one should draw per hour of flight time. The number of maneuvers per hour of flight, denoted n_{mpf} was chosen in order to match the overall severity of the flights in terms of the load cycle exceedance curve and crack growth.

3.2 Steady Flight Loads

The previous discussion has detailed a process for identifying maneuvers in flight recorder data and fitting a statistical model to the sequential maneuver information. This comprises the right-hand side of Figure 1. Next, the discussion will move to identifying steady flight portions of the flight recorder data and fitting a statistical model to them, as shown on the left side of Figure 1.

First one must identify which time hacks in the flight recorder data comprise steady flight behavior. To do this, another maneuver is added to the maneuver dictionary, the steady-flight maneuver (the 111th maneuver). Steady flight will be matched whenever the roll, pitch, and yaw rates are near zero and NNZCG < 4 g. Steady flight can occur at any Mach number and altitude.

Again, this definition can easily be modified and the subsequent analysis re-run in the future. The maneuver identification is done exactly as described above, but with the added steady flight maneuver. Any time hacks that are identified as steady flight are then removed from the maneuver analysis and considered separately.

The steady flight time hacks are analyzed to extract the load cycles that occur in steady flight. Each set of sequential steady flight time hacks is considered an example of steady flight behavior. Since the flight simulator does not simulate steady flight, the only source of loading information during steady flight is from the processed loads in the flight recorder data.

A first pass at the steady flight model drew samples from the overall distribution of steady loads to reach a (random) number of extrema. In comparing exceedance curves, this model resulted in widely separated straight lines for the low-load portion of the spectrum. In fact, it can be shown that simply drawing random loads from a distribution will generally result in straight lines on an exceedance plot. However, the actual flight recorder data shows distinct curvature in the low-load portion of the exceedance curve. In addition, the variation in the overall steady loads was high compared to the variation among flights. Thus, a more accurate steady flight model should result in some curvature in the exceedance curve and have reduced variation compared to the original model.

In order to investigate various steady flight modeling options, a new tool was developed. One way to more closely match a desired exceedance curve is to draw cycles from a known cycle distribution (rather than load points). One must be able to then generate a load history from a set of cycles. This procedure is not exact, since many load histories can have the exact same cycle content. Still, such a procedure can be developed [4] which returns a particular sample load history with the correct cycle content. However, another obstacle appears because some sets of cycles have no possible load history. One way to fix this is to add an artificial cycle to encompass the smallest and largest load values, but this large cycle is not physical and imparts error into the result. A better option is to sequentially draw cycles, where at each step the pool of cycles to draw from is restricted to those that *can* be inserted into the current load history. We take the second approach.

Given this capability, there are many options for drawing random cycles and including them in the load history. First, steady cycles must be extracted from the flight recorder data. The maneuver identification defines time hacks which are considered to be in steady flight. Cycles then can be extracted by either concatenating all of the steady time hacks together, or else by considering each steady maneuver (sequence of steady time hacks separated by maneuvers) separately. These two methods create different pools of steady cycles from which to draw when creating random steady loads. When drawing from the cycles, one can constrain the overall number of cycles to match the number in the flight recorder data, constrain the steady maneuvers to be the same as the original data, or have the same number of cycles but different content.

There are many options for precisely how to construct the steady loads, and around 8 different options were investigated. Many produced very wide distributions or did not match the exceedance curve well; of course, some options reduce to exactly matching the exceedance curve by introducing essentially no randomness. In the end, the method to use was chosen to have moderate variation in the resulting steady loads and to be easy for the Air Force to modify during the test execution.

The final method is as follows:

1. Concatenate all steady time hacks together and perform cycle-counting. This returns the pool of steady cycles encountered in the flight as if no maneuvers had been performed. This results in n_{steady}steady cycles.
2. Allocate $n_{steady}/n_{maneuver}$ cycles in between each maneuver, where $n_{maneuver}$ is the number of maneuvers.
3. Between maneuvers, draw cycles from the overall pool randomly (with replacement) to reach a total of $n_{steady}/n_{maneuver}$. At each draw, only draw from those cycles which can be inserted into the current set of cycles to generate a physical load history. This results in a steady load history between each maneuver.
4. Concatenate the steady load histories with the maneuver load histories from the flight simulator. This results in the final load history for the entire flight.

This process results in a meaningful set of steady flight data for the probabilistic load spectrum. Since each piece of steady flight is treated differently, it can be generated independently and inserted at will in during the flight. This gives the Air Force flexibility to modify the test loads as desired.

At this point, each of the processes depicted in Figure 1 have been described. For clarity, the result of this process is a probabilistic description of the load spectrum which may be drawn from to simulate aircraft loading. This description consists of the Markov transition matrix with the associated loads for each maneuver (Stick-to-Stress output files) and the load cycles for steady flight. Given this information, one can draw a sequence of maneuvers from the Markov transition matrix and produce a maneuver history. Then, one can place a steady flight history in between each maneuver, with content drawn from the steady load cycles. The assembled load history is a sample from the probabilistic load spectrum. The load history can be cycle-counted to display its exceedance or occurrence curves, or passed into a crack growth calculation.

3.3 Demonstration with Flight Records and Flight Simulator

The results of the maneuver identification and clustering are shown in Figure 2. The vertical axis is the maneuver ID, which runs from 1 to 111 (the figure is zoomed in) and the horizontal axis is time in seconds. The blue dots show which maneuver is identified at each time hack. There are four clear clusters, each of which corresponds to a maneuver. The four maneuvers are identified by the clustering algorithm, shown as red circles. The clustering clearly groups the maneuvers together while filtering out the noise, i.e. the occasional aberrant time hack.

Next, the Markov transition model is fit to the sequence of maneuvers (one per cluster). The transition matrix is shown in Figure 3. As one might expect, not all 110 maneuvers occurred in the single flight record that was analyzed, so some rows and columns are all zero (white). The transition matrix is well behaved and can be used for drawing maneuver sequences (Figure 3).

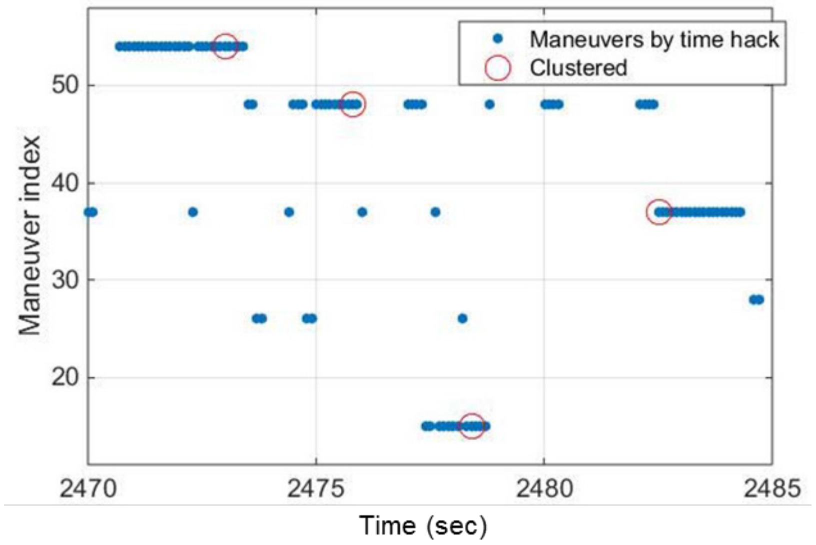


Figure 2. Result of Maneuver Identification and Clustering.

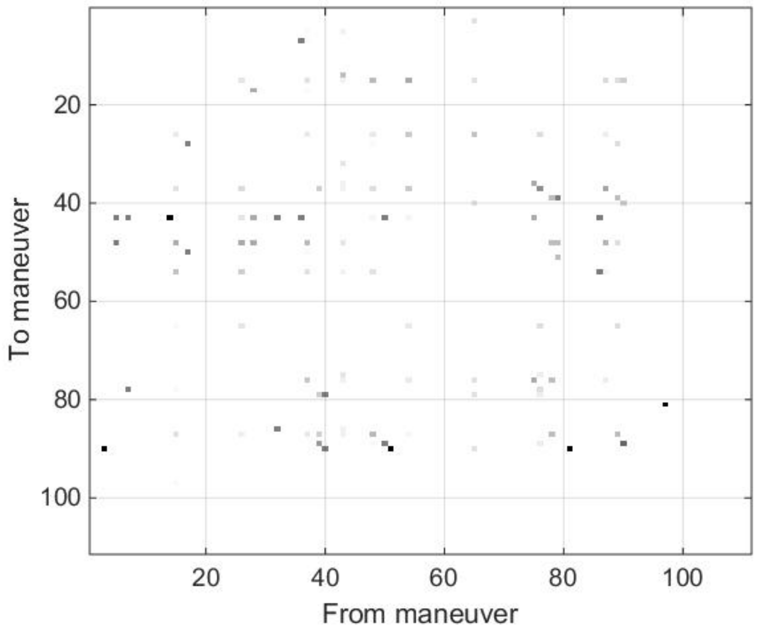


Figure 3. Sample Markov Transition Matrix.
Darker points indicate higher probability transitions.

Finally, with all of the probabilistic models fit, we draw a simulated load history from the probabilistic load spectrum. An example is shown in Figure 5. The maneuvers are clearly identified as smooth portions of the history, whereas the steady flight portions between maneuvers is only a set of extrema which correspond to the steady load cycles. Of course, this load history will be processed by finding extrema and counting load cycles, so the time axis is not particularly meaningful. In any case, the load history looks reasonable.

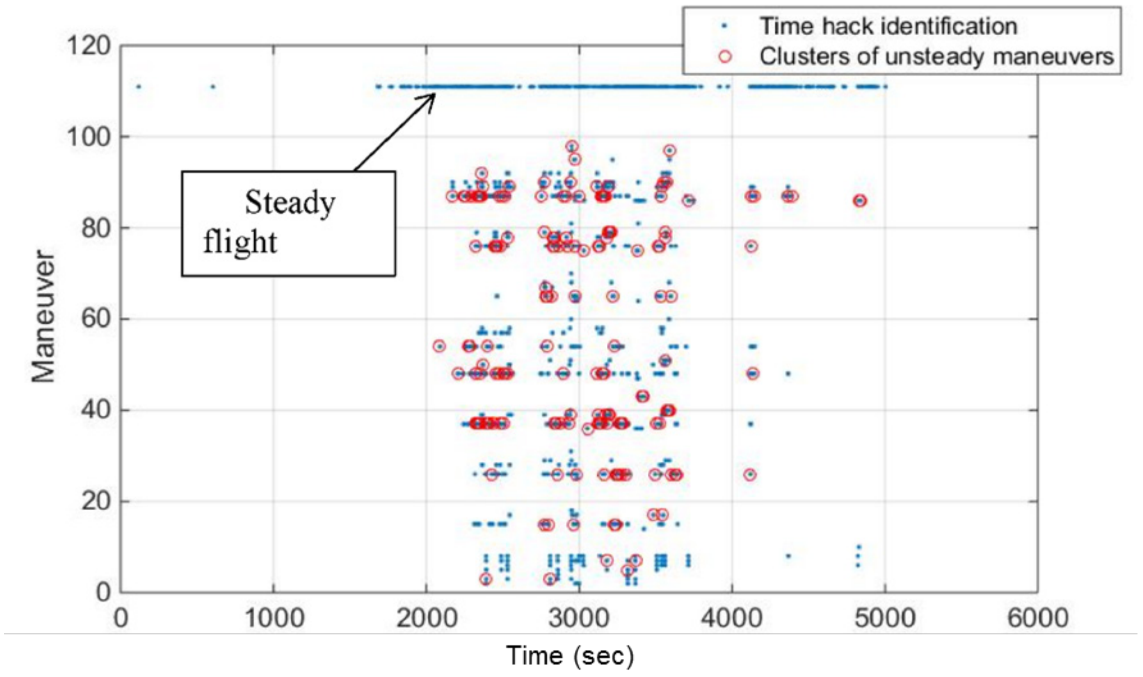


Figure 4. Full Maneuver Identification, Including Steady Flight Maneuver.

Finally, the exceedance curves of the simulated and actual flight histories are compared. In this comparison, the histories are first converted to extrema, then edited at a 5% truncation level using the Lockheed-Martin truncation algorithm to remove small noise cycles, and finally cycle-counted. The resulting exceedance curves for load cycle amplitude for 20 samples from the probabilistic load spectrum are shown in Figure 6. The value of n_{mpf} was set to 20 maneuvers per flight hour to get the sample exceedance curves to approximate the actual curve. Increasing n_{mpf} essentially scales the sample curves up. In the final probabilistic load spectrum, n_{mpf} is adjusted to match the damage induced by real aircraft actual loads and is set to different values for each mission type. The sample exceedance curves tend to encompass the actual data, giving good confidence in the method of designing the probabilistic load spectrum. The bottom right of the curve comprises high-load cycles due to maneuvers which vary among the simulated flights. The sample curves are choppy here because the maneuvers in the flight simulator have a wide range of cycle amplitudes and the bottom right represents the rarest load cycles. The upper left portion of the curves is mostly due to the steady flight model and consists of more frequent, lower amplitude cycles. This region generally captures the trend of the data, though the slope is nearly constant. In fact, the steady flight model used in these plots was a simpler one based on drawing steady flight load points rather than load cycles. The final algorithm draws load cycles and more correctly captures the curvature in the low-load region of the exceedance curve.

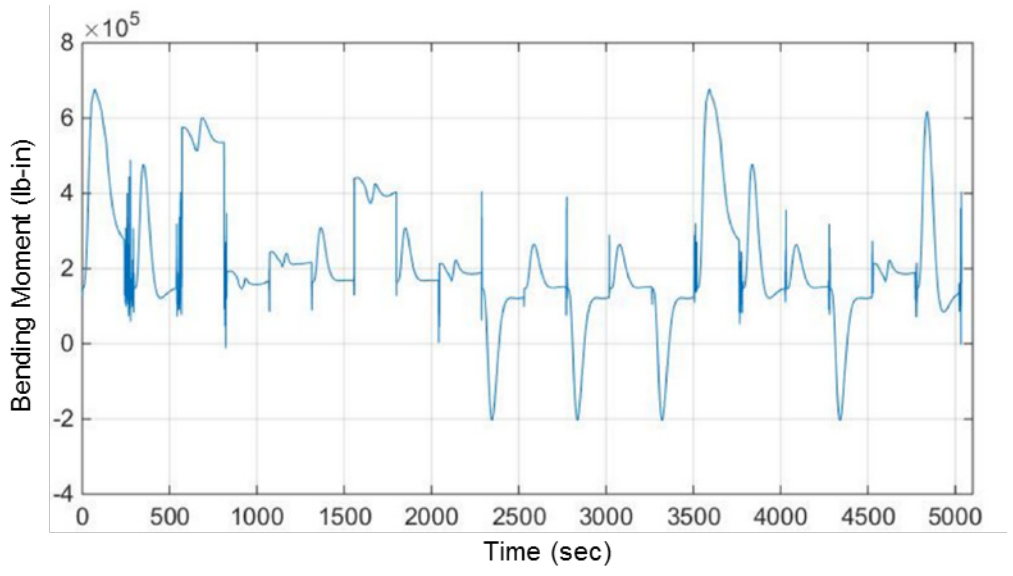


Figure 5. Draw from the Probabilistic Load Spectrum Showing Load History.

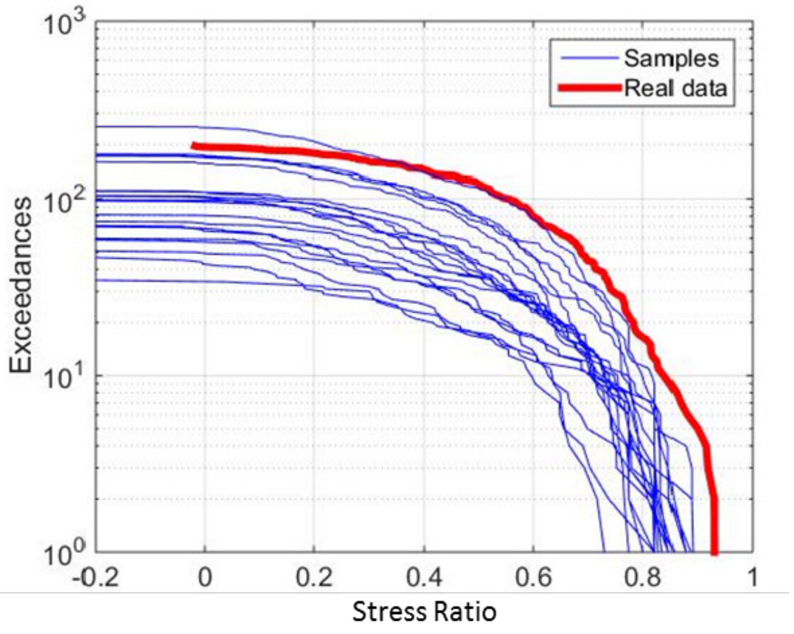
Figure 6a shows the exceedance curve for the stress ratio. The plot shows that the samples generally have somewhat lower stress ratio that the actual data. This is to be expected, since the team has found that maneuvers generated by the flight simulator have consistently lower stress ratio content than flight recorder data. In addition, it was suggested that the lower stress ratios may be more realistic. In this case the trend is in fact a good sign that the probabilistic load spectrum is giving realistic simulated flights.

The load cycle amplitude exceedances of the samples cluster around the exceedances from the actual data in Figure 6b. At low amplitudes, the samples tend to have fewer exceedances than the actual data. This is not a significant concern since these low amplitude cycles are not very damaging. The samples have more high cycle amplitude exceedances than the actual data. This is a concern and was the motivation for work discussed in the next section.

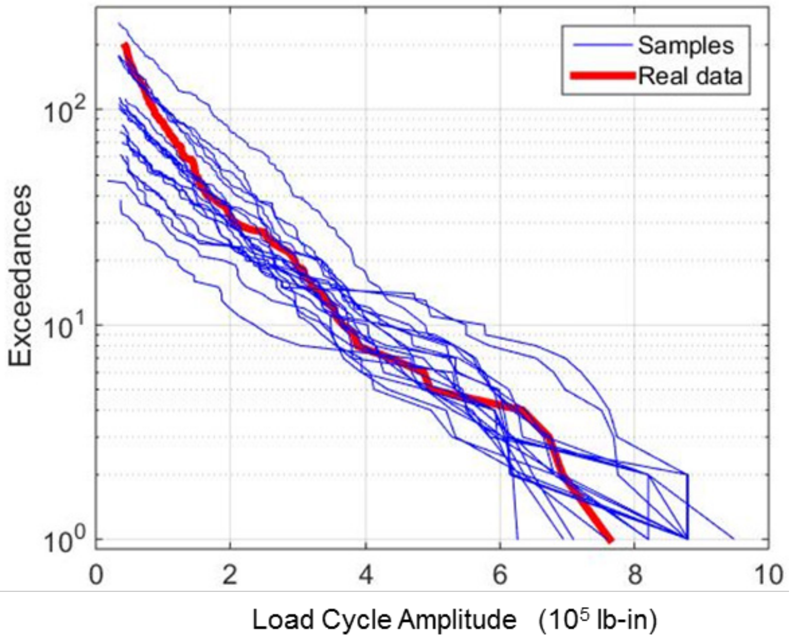
Note that the maneuvers inserted into a synthetic flight are randomly flipped from a right-hand maneuver to a left-hand one. This is because the asymmetric maneuvers in the maneuver database are all right-turning maneuvers. When a maneuver is flipped, the loads on the left and right wings are swapped.

3.4 Reordering Cycle Content

The exceedance curve was used as the basis of comparison due to its simplicity and speed of calculation. However, in reality the metric of interest to evaluate the probabilistic load spectrum is actual crack growth. Thus, the current probabilistic load spectrum was run through crack growth for 1000 flights with all other parameters set to nominal values (material properties, external-to-internal load and stress evaluation, etc.). For comparison, the actual flight records (or the post-processed loads provided to GE) were run through the crack growth routine as well. To evaluate the spread in the probabilistic load spectrum, another comparison was done by taking the actual flight loads and re-ordering the cycle content. This is akin to looking at the same set of maneuvers but performed in a different order. It is presumed that the variation resulting from the ordering of maneuvers should be less than the variation in the overall load spectrum, which should include variation due to the pilot, seasonality, mission, etc. However, the variation in the probabilistic load spectrum should not be many orders of magnitude larger than the variation due to cycle ordering.



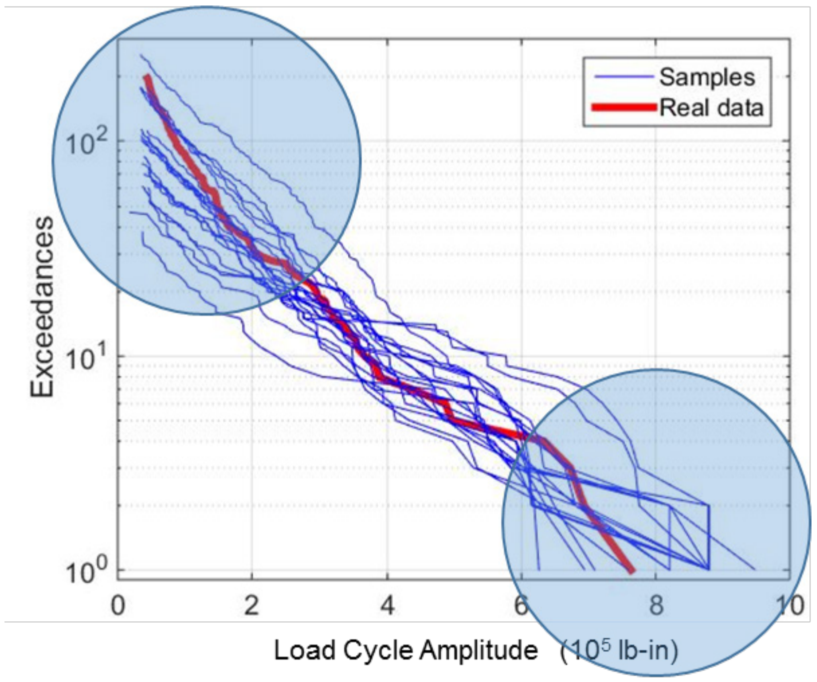
a) Stress Ratio Exceedances



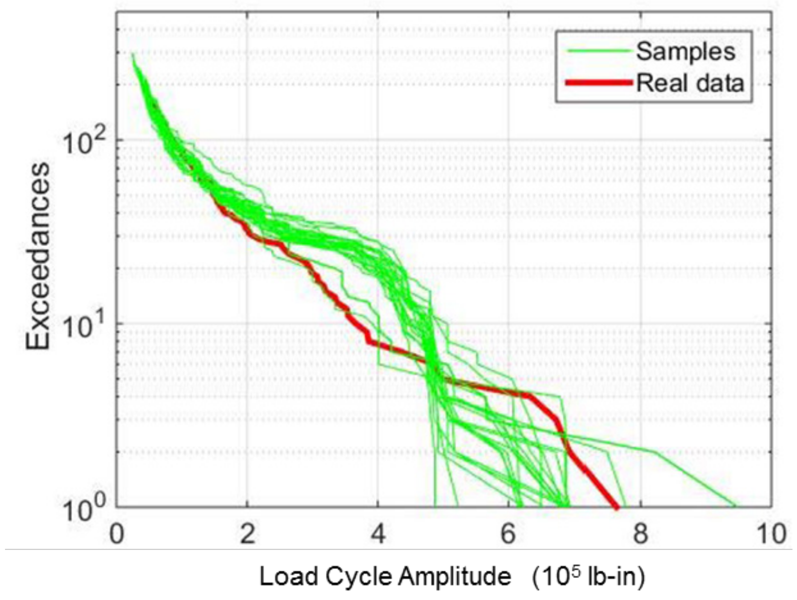
b) Load Amplitude Exceedances

Figure 6. Exceedance Curve Comparison of Samples from the Probabilistic Load Spectrum and the Sample Flight Histories.

The effect of the steady flight model on the exceedance curve is shown in Figure 7. In Figure 7a, the upper left circle identifies the portion due to the steady flight model, while the lower right circle identifies the portion due to maneuvers. Figure 7a clearly shows the problem with a simple steady flight model is that it has wide variation and does not capture the curvature of the exceedance curve. Figure 7b shows that the final model has a reduced (though not zero) variation and follows the overall curvature of the exceedance curve. Some mismatch occurs in the moderate load region and is due to the way in which the maneuver and steady loads are combined. Removing the constraint that the test loads must be easily modifiable by the Air Force may allow this discrepancy to be reduced, but the problem is not a serious one. In the end, it is the crack growth which matters, and the probabilistic load spectrum need not match flight recorder data exactly.



a) Simple Steady Flight Model



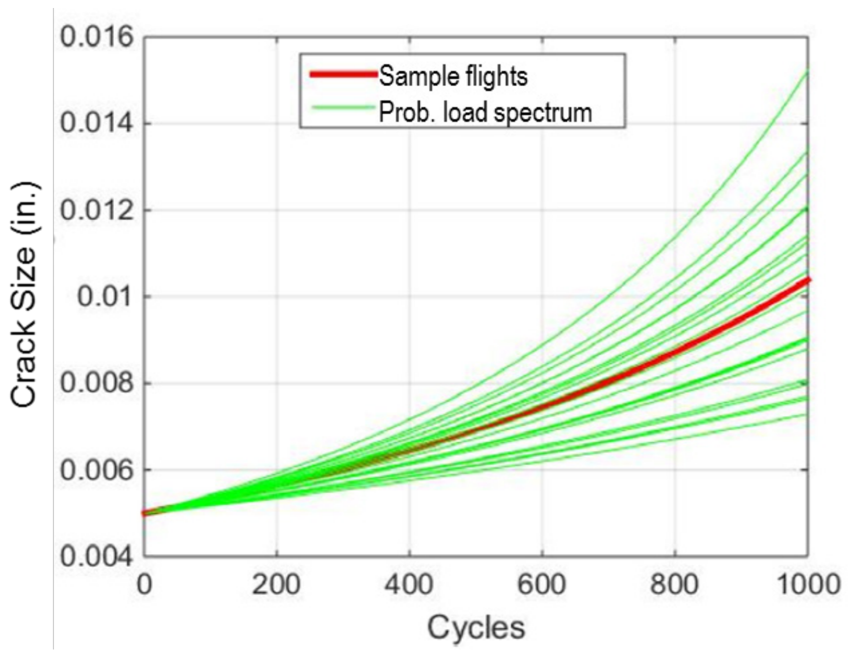
b) Random Draw Steady Flight Model

Figure 7. Probabilistic Load Spectrum with Simple Steady Flight Model and Final Version Where Load Cycles are Drawn.

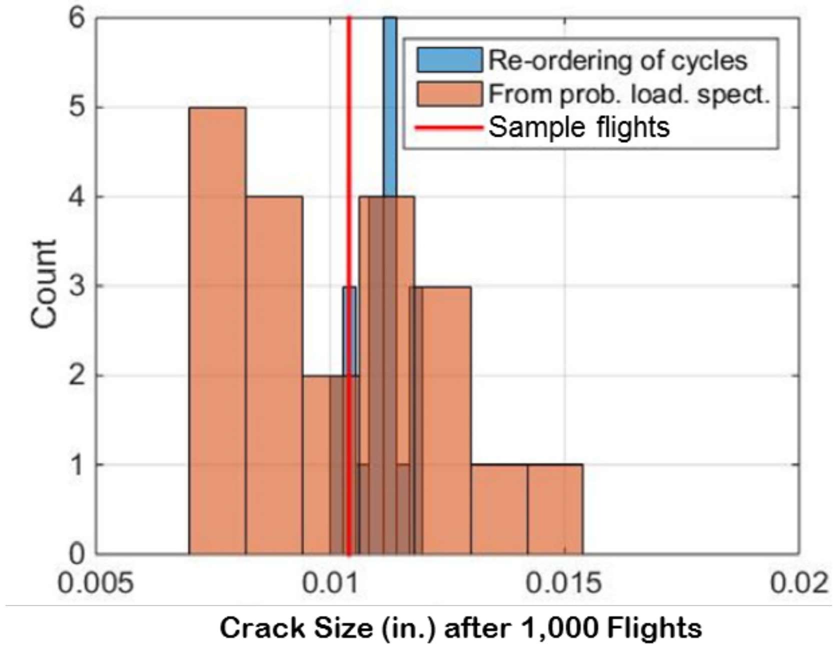
3.5 Crack Growth Comparisons

Crack growth comparisons are shown in Figure 8. The crack growth from the probabilistic load spectrum is shown in Figure 8a to be centered around the crack growth predicted by the sample flight records, with reasonable variation. Figure 8b shows somewhat larger variation in the crack size after 1,000 flights from the probabilistic load spectrum compared with re-ordering the cycles in the sample flight records, as expected. Some other steady flight models that were investigated had considerably larger variation (multiple orders of magnitude) and were therefore discarded.

In order to reduce the file size of the final master event sequence (MES), it is passed through an additional step of extrema finding. A load point is retained if it corresponds to an extremum in any of the external load parameters.



a) Crack Growth Curve



b) Histogram of Crack Size after 1,000 Flights

Figure 8. Crack Growth Predictions from the Probabilistic Load Spectrum.

3.6 Up- and Down-Bending Maneuvers

The flight records had a relatively small number of maneuvers occurring with down-bending. While down-bending is rare, it needs to be included in the test as it is an important test of the limits of the modeling approach. Currently, the probabilistic load spectrum is being used to design a test plan, and so all possible load cases must be considered. Thus, the probabilistic load spectrum had to be modified to include more down-bending maneuvers.

The process of adding down-bending maneuvers is actually quite simple. First, down-bending maneuvers are identified in the Stick-to-Stress runs. Of the 110 Stick-to-Stress runs that server as the database, two had major down-bending, -2g and +8g symmetric maneuvers. The wing bending moment time histories for these maneuvers are shown in **Error! Reference source not found.**. Note that the +8g pull up has some large oscillations which lead to large negative bending moments, even though the peak bending moment is positive. Once the maneuvers with negative bending moment are identified, the transition matrix is modified to include more transitions from and to these maneuvers. The relative frequency of the maneuvers occurring can be controlled by a scale factor on the entries in the transition matrix. For the purposes of designing the test, the factor was tuned to result in about one down-bending maneuver per flight.

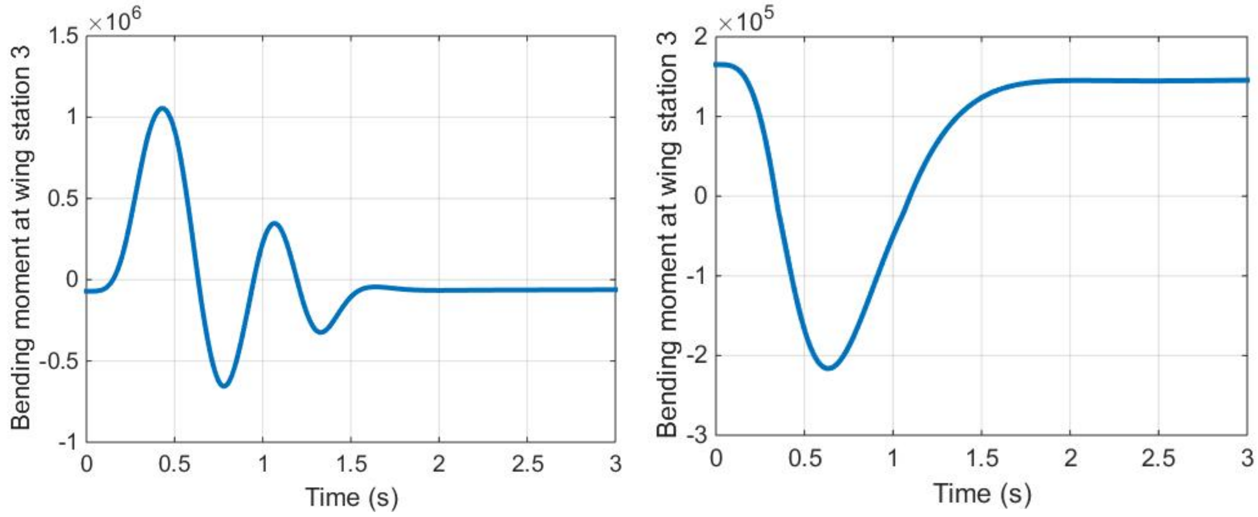


Figure 9: Bending moment at wing station 3 time histories for +8g pull up (left) and -2g push down (right) maneuvers as calculated by the flight simulator.

3.7 Estimated Flight Recorder Data, Ground Loads, and Altitude

The probabilistic load spectrum is divided into steady-flight portions and maneuver portions. The steady flight portion is modeled by selecting "steady" load cycles. Since these cycles are not associated with flight recorder data (FRD), it must be estimated. The flight simulator database provides the necessary link. However, the solution is not uniquely defined. That is, there may be multiple PITS (Mach, altitude, roll and pitch rates and accelerations) that could result in the same loading condition. Our approach is to use GE's Bayesian Hybrid Modeling (GEBHM) tool to model loads as a function of FRD, exactly as will be required while running P^2IAT. Then, this model is essentially inverted so that FRD is calculated given a set of loads. As the solution is not unique, the inversion process takes the FRD which is closest to one of the initial data points used to train the BHM model.

Altitude during the steady flight periods is assumed to connect the maneuvers. That is, each maneuver occurs at a known altitude, and the steady flight period is meant to model the transition from one maneuver to the next. Thus, the altitude during the steady flight should transition from the previous maneuver to the next. The ground-air-ground cycle is modeled by adding a "ground" data point (i.e. FRD and loads) at the beginning and end of each flight. Note that GE and the Air Force have agreed not to model landing or taxi loads, so the final ground data point is essentially the same as the initial one having Nz equal to 1. Ground loading is drawn from the flight data. For each of the 12 flights, the loads at the lowest altitude point are extracted as the ground loads. It was verified that these do indeed correspond to points on the ground. The loads are generally small, e.g. -1,000 lb-in bending compared to a typical maneuver value of 50,000 lb-in. The flight recorder data at the ground point is estimated such that the GEBHM model which converts flight recorder data to loads will produce the correct loads. That is, the BHM model is inverted so that FRD is calculated for the known ground loads.

3.8 Grouping into Missions

Five missions were developed for the baseline usage spectrum. Since the flight records delivered by the Air Force had only a training mission and an unknown mission reported, the missions were not developed to fit directly to real missions. Rather, the load spectrum was generated with the aggregated data and missions were generated that correspond to subsets of the overall variation that was encountered. For example, the easy mission corresponds roughly to the lower 25% of the overall flight severities encountered in the flight records.

Each mission is described by a maneuver transition matrix which encodes how often each maneuver occurs, as well as a scalar value for the number of maneuvers per flight. The five missions, developed by hand, are as follows:

1. Easy – consists of low-g, low-altitude, low-Mach number maneuvers.
2. Baseline – covers the entire range of the given flight records.
3. Short hot – small number of high-g maneuvers.
4. Moderate – many medium maneuvers, preferring low-g.
5. Extreme – many high-g maneuvers.

The mission mix is then defined which describes how often a flight from each mission will occur. Generally, the more extreme missions occur less often. The mission mix can easily be modified by the Air Force and P^{2}IAT should be able to quickly learn a new mission mix. The mission mix allows P^{2}IAT to adapt to how a given asset is operated. The most up-to-date mission mix, specific to each asset, is used to fill in missing data and to forecast future usage.

The probabilistic load spectrum missions are defined by a maneuver transition matrix, which describes how often maneuvers occur and their sequential information. The transition matrices for each mission are shown in Figure 10 through Figure 14. Darker color indicates a more likely maneuver or transition. Note that the only difference between the short hot mission and the extreme mission is the number of maneuvers per flight; the transition matrices are the same.

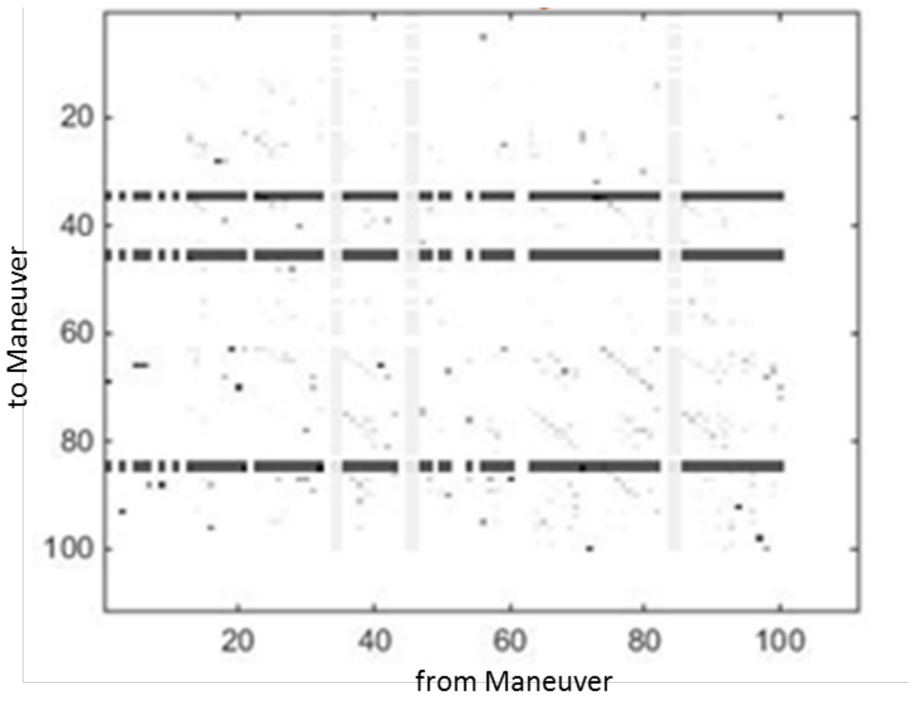


Figure 10. Maneuver Transition Matrix for Easy Mission.

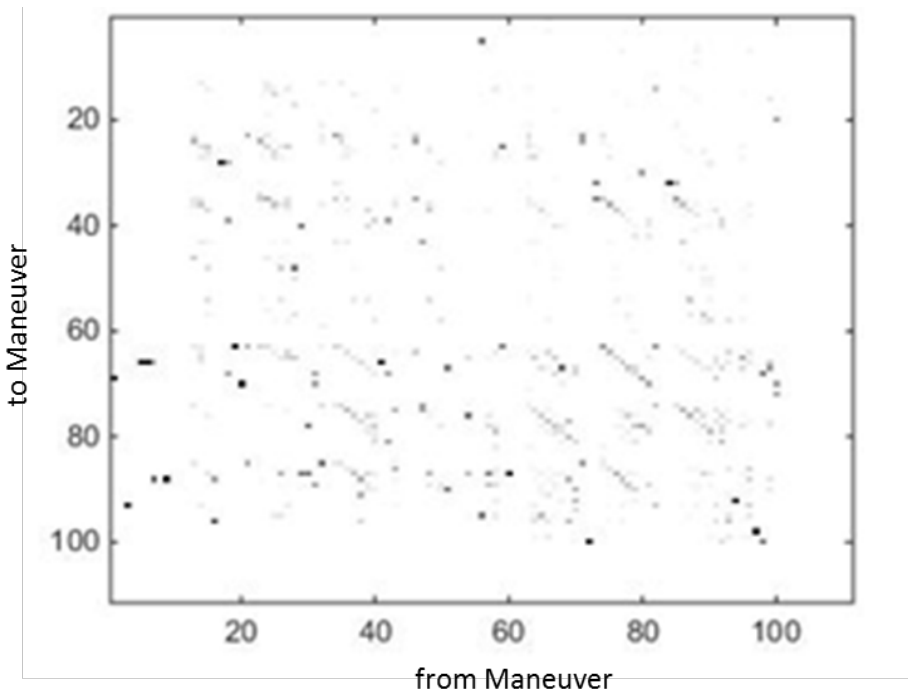


Figure 11. Maneuver Transition Matrix for Baseline Mission.

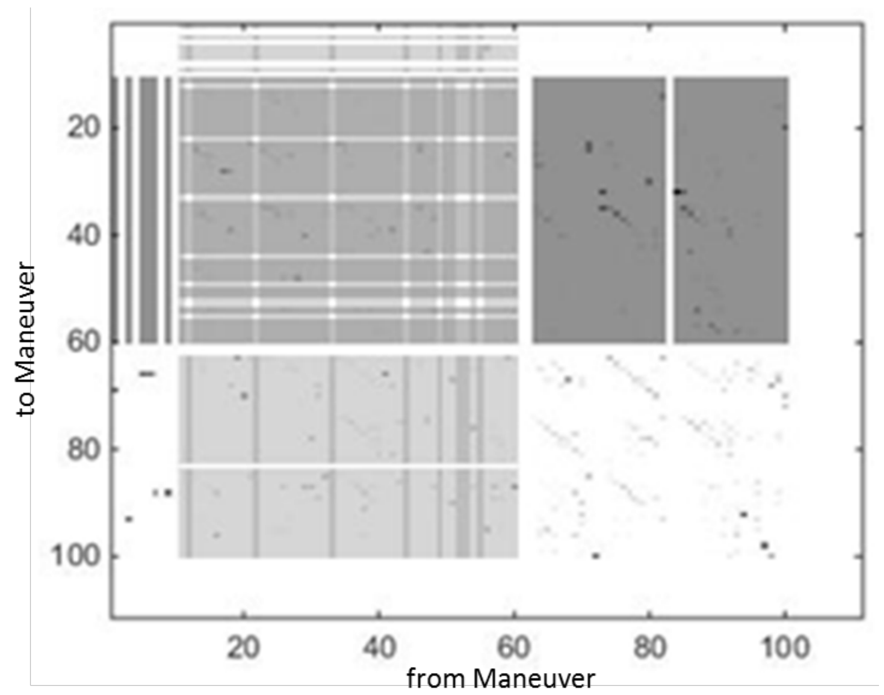


Figure 12. Maneuver Transition Matrix for Short Hot Mission.

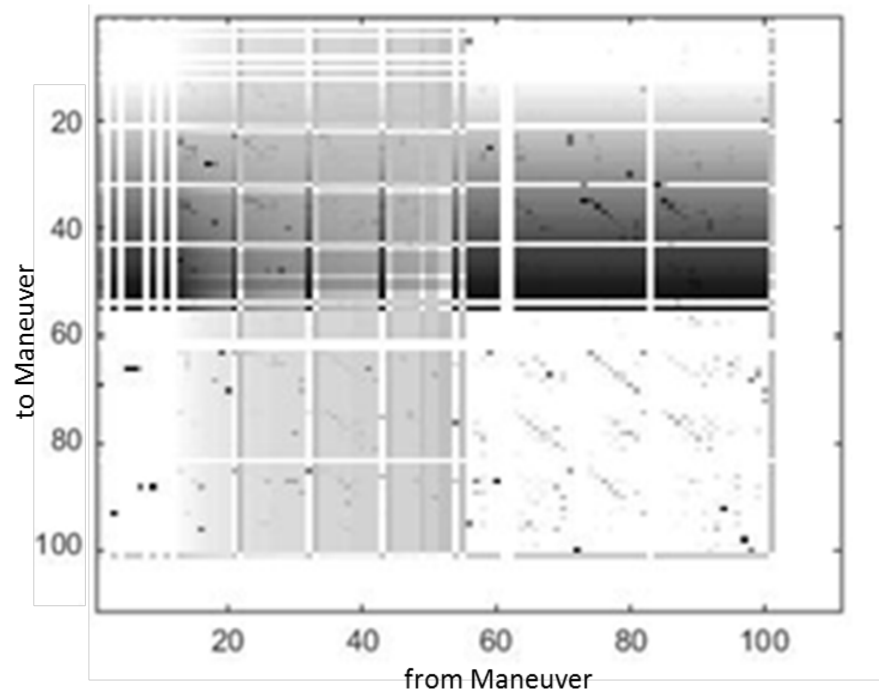


Figure 13. Maneuver Transition Matrix for Moderate Mission.

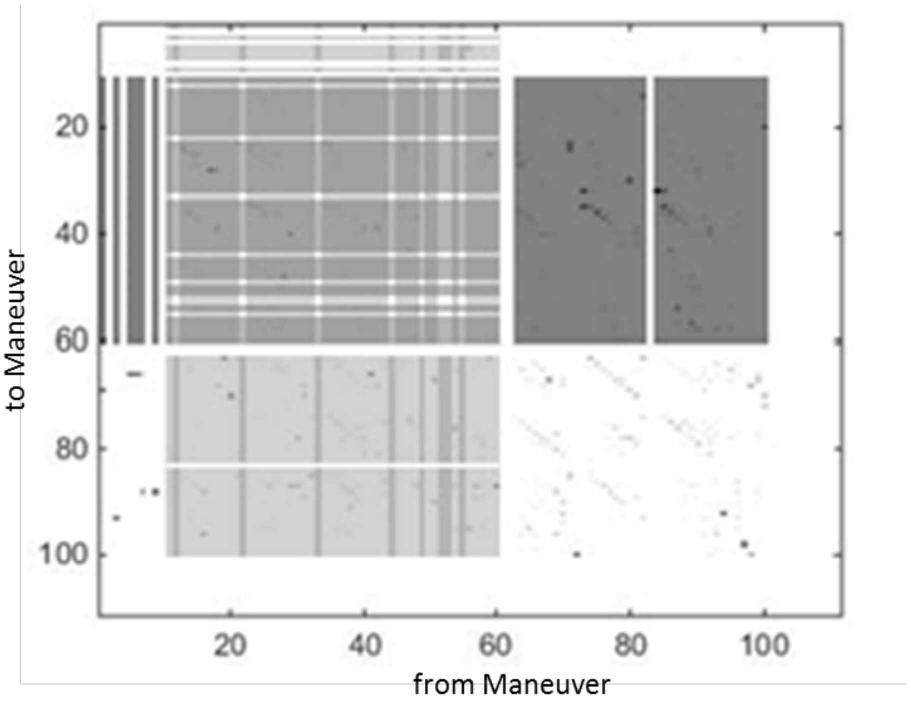


Figure 14. Maneuver Transition Matrix for Extreme Mission.

One thousand flights were drawn from the probabilistic load spectrum with the mission mix shown in Table 1.

Table 1. Mission Mix.

Easy	Baseline	Short hot	Moderate	Extreme
0.2	0.3	0.2	0.2	0.1

The combined one thousand flights (each with an associated mission) are referred to as the MES, the probabilistic load spectrum, or the baseline usage data. Exceedance curves for the outboard bending moment and Nz are shown in Figure 15 and Figure 16, respectively. The MES is also compared to usage from other similar aircraft in Appendix C. The entire MES is provided in Appendix H in volume 2 of this report.

One important uncertainty in modeling asset usage is pilot-to-pilot variation. In P^{2}IAT, this is considered by modeling variation between different instances of the same mission type. Pilot-to-pilot variation could show up in, for example, slightly different maneuvers being performed for a given mission. This variation is accounted for in the probabilistic model of the missions, and the probabilities are fit using flight records. Thus, pilot-to-pilot variation in the actual flight records is modelled in the final MES. In particular, drawing multiple different flights with the same mission definition represents how different pilots might fly the same mission. Thus, pilot-to-pilot variation is included in the probabilistic load spectrum by having multiple realizations of each mission.

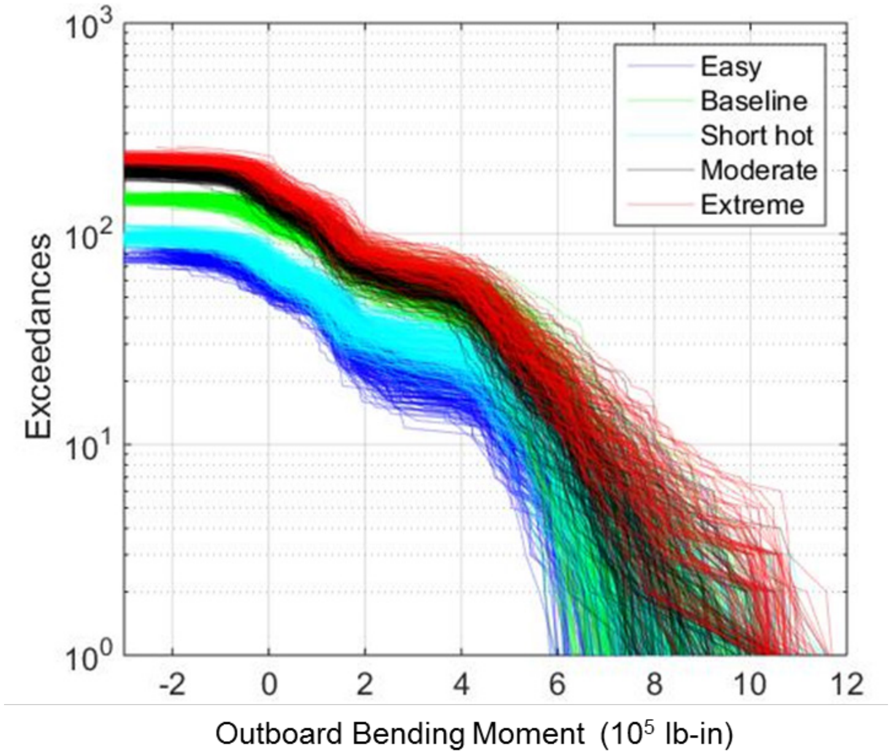


Figure 15. Bending Moment Exceedance Curves for Missions in the MES.

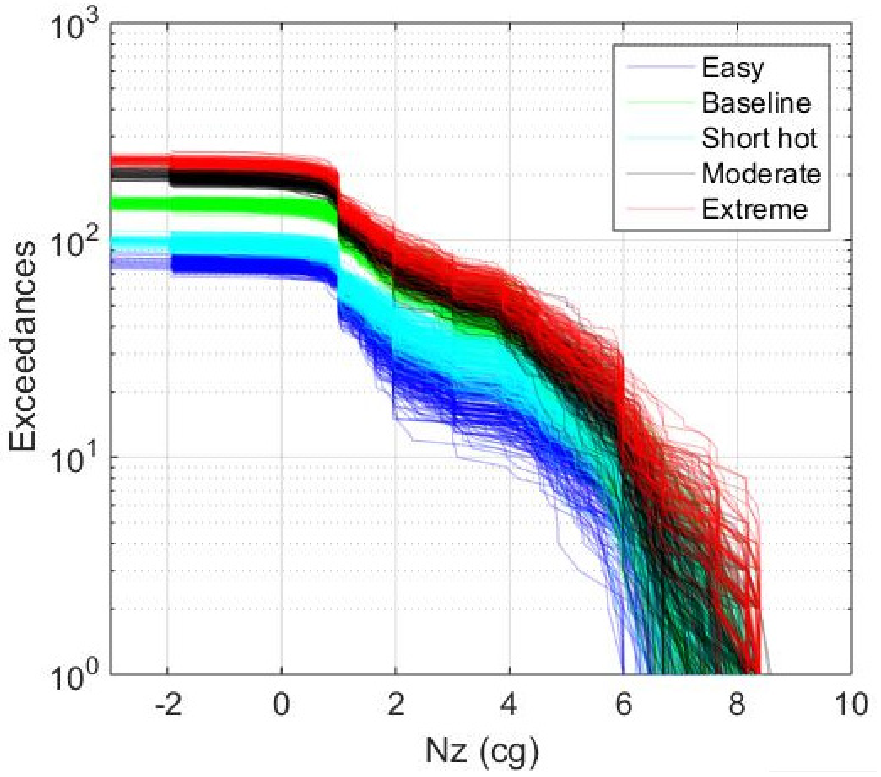


Figure 16. Nz Exceedance Curves for Missions in the MES.

3.9 Gross Weight Considerations

For this effort, the gross weight of the aircraft was kept constant at 40,000 lb.

3.10 Incorporating into P^{2}IAT

The MES is used in a number of ways in P^{2}IAT. Now that a baseline MES has been generated, it is important to verify that P^{2}IAT can use it as input for each of these purposes. First, the MES is used as the actual flights and missions flown (i.e. flight data). The baseline life calculations are done assuming that the load spectrum is the actual flight data. Second, the mission data are used to learn the mission mix. Third, the MES is used as a basis for forecasting. Forecasting is done with the learned mission mix and results in crack growth over time with post-processing that can identify time to failure, time to inspect, etc. Fourth, the MES, with some portions missing to simulate missing data, is passed through two stages before it is used as loading:

- Fill in missing flight recorder data using correlated normal assumption (more complex and accurate methods are possible but will take some time to implement).
- Fill in missing flights using a given mission ID or, if none is given, draw a mission ID from the current estimate of the mission mix.
- Estimate loads from flight parameters (flight recorder data) using a pre-built BHM model which includes prediction uncertainty.

All of the above methods were implemented in P^{2}IAT. The missing flight recorder data filling and load prediction from flight parameters are performed in a pre-processing step to improve computational efficiency. Code was also written to draw from the probabilistic load spectrum when doing forecasting. Note that these draws are from the 1000 flights in the MES; no new synthetic flights are generated as P^{2}IAT is executed.

4.0 SAFER-P^{2}IAT

The SAFER- P^{2}IAT framework takes as input several data sources and models. These data and models describe the materials, geometry, structure, operational usage, inspections, and failure criteria of an operating aircraft, or in general any asset. The framework combines all this data to compute damage progression and risk of failure over time. The framework also forecasts damage into the future, predicts inspection times and results, and can schedule inspections based on a number of rules.

Defining some of the inputs requires non-recurring engineering work. These inputs include data and models of the material, geometry, and structure (e.g. finite element models). Inputs which are taken from the field include operational usage and inspection data. With appropriate engineering assumptions, some inputs may be simply taken from past studies without modification. These may include, for example, fatigue crack growth curves or Probability of Detection curves.

The flow chart in Figure 17 shows the SAFER- P^{2}IAT analysis process. Blue boxes represent data sets that are either input or calculated. Green boxes represent models that must be built as a part of the non-recurring engineering work for a given control point and asset. Note that the boxes may also be referred to as nodes. The framework is built so that data or uncertain distributions may be supplied for most of the blue boxes. If no data are supplied for the flight recorder data, mission data, detect/no-detect and fail/no-fail nodes, then those will be estimated. Other data sources (e.g. geometry, initial crack state, etc.) must be given either a fixed value or a probability distribution.

The overall process in Figure 17 remains the same whether one is performing a data-assimilation or forecasting task. When doing data-assimilation, FRD and mission data are supplied with possible missing data. Inspection data may also be supplied to the detect/no-detect node. The result of data-assimilation is a current estimate (diagnosis) of the crack state probability distribution. Additional outputs are updated distributions for all uncertain parameters, including the mission mix (probability of occurrence of each mission type).

When doing forecasting, the current crack state is projected forward in time using a load forecasting model. Load forecasting is done by drawing flights from the baseline usage data (alternatively referred to as the MES or the probabilistic load spectrum) according to a current estimate of the mission mix. The forecasted loads then drive the fatigue crack growth calculations. Inspection scheduling can be based on a fixed interval, an SFPOF criteria, or an information gain criteria.

Many of the nodes (data and model) in the flow chart in Figure 17 must be defined for a given control point and asset (as mentioned above, some may be omitted for certain types of analysis). The following sections describe the data and analyses used to generate the nodes for tracking/predicting crack growth on an airframe component. Later sections describe the particular data sets and results for a single control point, the demonstration location (crack at a bolt hole), on the outer wing of a fighter aircraft.

At this point it is worth describing how the flow chart in Figure 17 is used to track crack growth, propagate uncertainty, and perform updating. In machine-learning terminology, the flow chart can be described as a Dynamic Bayesian Network (DBN) which can be analyzed using various filtering methods [5]. A DBN is essentially a set of variables and models which compute outputs (e.g. probability of failure) from inputs (e.g. flight data, material properties, etc). The computations are arranged in a network which describes the input-output relationships of the various models. The DBN is dynamic because the crack length is grown over time as the data from each flight is sent into the network; this is the how the crack length is tracked over time. The DBN is Bayesian because a Bayesian method (sequential importance sampling) is used to update the uncertainty using inspection data.

In P^2IAT, the DBN is analyzed using a Particle Filter (PF) method [6]. Some of inputs to the flow chart are uncertain quantities (e.g. material properties). Uncertainty is propagated by evaluating the network (i.e. performing the crack growth analysis) for various settings of the uncertain quantities. The result is a description of the probability distributions of the input and output variables. The distributions can be updated using an application of Bayes' rule when measurement or inspection data are available. In this way, updated predictive distributions of outputs such as crack length are available at every flight. The next sections will provide more details on the algorithms used to propagate and update the uncertainty in the network. Following this, the assumptions and models required for tracking crack growth in a fighter aircraft (i.e. the boxes in Figure 17) are discussed in detail.

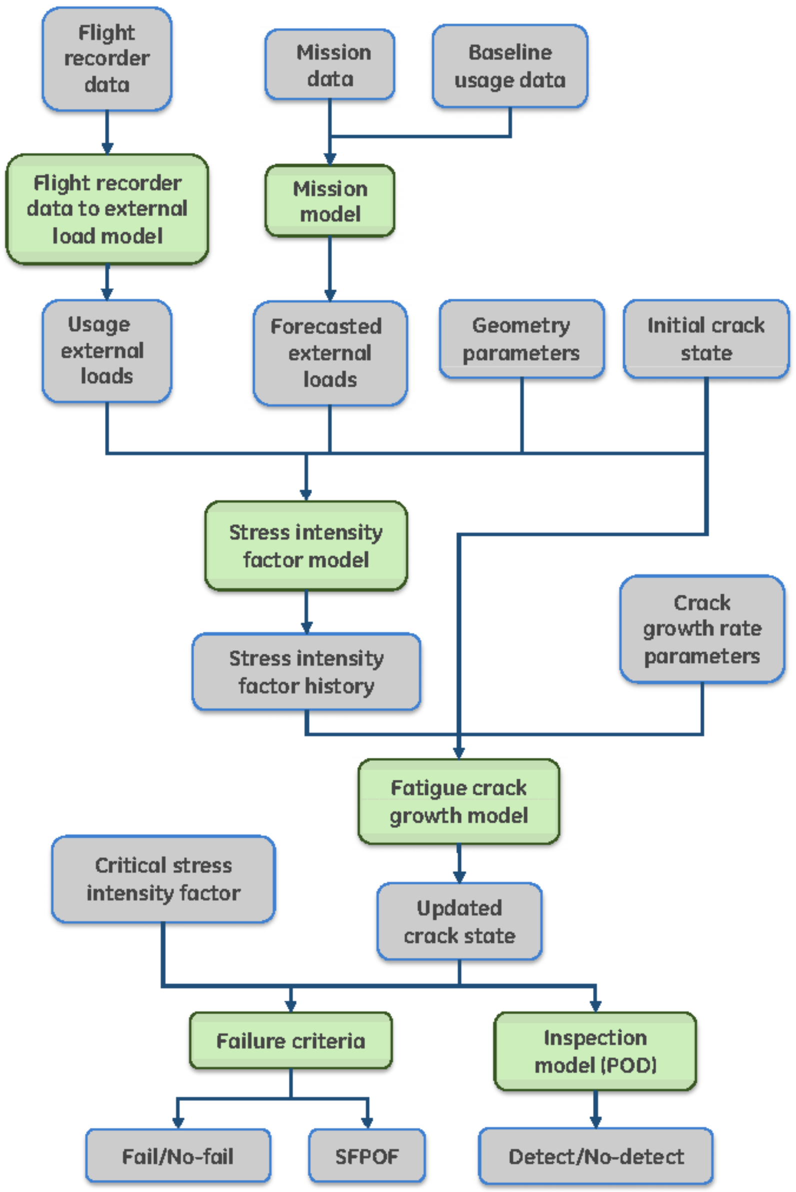

Flight recorder data
Mission data
Baseline usage data
Flight recorder data to external load model
Mission model
Usage external loads
Forecasted external loads
Geometry parameters
Initial crack state
Stress intensity factor model
Crack growth rate parameters
Stress intensity factor history
Fatigue crack growth model
Critical stress intensity factor
Updated crack state
Failure criteria
Inspection model (POD)
Fail/No-fail
SFPOF
Detect/No-detect

Figure 17. SAFER-P^{2}IAT Analysis Flow Chart.

4.1 Uncertainty Propagation

In P^{2}IAT, uncertainty propagation is performed using a Particle Filter method. The modules and data which are the constituent parts of P^{2}IAT are combined into a network as shown in Figure 17. Since a probabilistic analysis is used to propagate and update uncertainty through the network, it is referred to as a Dynamic Bayesian Network (DBN) [7]. A DBN can be considered as a series of BNs (Bayesian Networks), one for each step in the data rhythm (either on a per-download basis or a per-flight basis). Based on the Markov assumption, the states of the current BN depend only on the BN at the previous time step and this dependence is generally independent of time [8]. In summary, the abilities to integrate various uncertainties and track system evolution over time make the DBN a suitable method for building the aircraft digital model.

A more concise and general representation of a Dynamic Bayesian Network is shown in Figure 18. The network consists of "hidden" or un-measurable states X, which evolve over time according to some relationship. The network can also predict measurable quantities Z at each time step. In analyzing the network, the uncertainty in X is initially described by a prior distribution, propagated through time using the network logic, and updated periodically using measurements of Z.

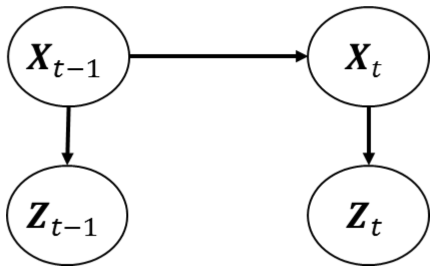


Figure 18. A Simple Dynamic Bayesian Network.

The DBN implemented in P²IAT is essentially the network shown in Figure 17. Each blue box represents a random variable (or set of variables) in the network. Green boxes represent deterministic transformations which are developed through engineering analysis. These are generally complex and nonlinear. Arriving at this particular network structure, which represents the linking of a suite of engineering analyses, was a major accomplishment of this work. While the structure is valid for many engineering assets, it may also be easily modified or extended within the SAFER-P²IAT framework. The structure itself, along with the definitions of every element within it, are part of the input specifications.

The Kalman filter gives an exact analytical solution for a linear-Gaussian DBN where all inputs are assumed to have Gaussian distributions and all models assumed linear. Extended Kalman filter and unscented Kalman filter (UKF) provide solution for a non-linear DBN by linearizing the state function to the first or second order, but they still assume that all the state variables are Gaussian. A generic DBN framework is needed for SAFER- P²IAT, which requires: 1) handling both discrete and continuous variables; 2) handling various types of continuous variables; 3) handling linear/non-linear functional relationships; and 4) no restrictive assumption of Gaussian distribution for noise term. As a sampling-based generic algorithm, the particle filter (PF), fulfills the requirements above and is chosen to update the DBN in our digital twin model.

A brief introduction to the PF method is given here [6]. In a simplest DBN in Equation 18, assuming that the state variables $\boldsymbol{X}_k \in \Re^m$ at time k evolves from the state variable $\boldsymbol{X}_{k-1} \in \Re^m$ according to:

$$\boldsymbol{X}_k = f(\boldsymbol{X}_{k-1}, \boldsymbol{v}_{k-1}) \tag{18}$$

and the measurement $\boldsymbol{Z}_k \in \Re^n$ is obtained according to:

$$\boldsymbol{Z}_k = h(\boldsymbol{X}_k, \boldsymbol{n}_k) \tag{19}$$

where $\boldsymbol{v}_{k-1} \in \Re^m$ and $\boldsymbol{n}_t \in \Re^n$ are vectors of noise terms in the evolution and measurement, correspondingly. The most basic particle filter algorithm is the sequential importance sampling (SIS) [9]. The SIS considers the full joint posterior distribution at time k, $p(\boldsymbol{X}_{0:k}|\boldsymbol{Z}_{1:k})$. This distribution is approximated with a weighted set of particles$\{\boldsymbol{X}_{0:k}^i, \omega_k^i\}_{i=1}^N$. These particles approximate the joint posterior distribution $p(\boldsymbol{X}_{0:k}|\boldsymbol{Z}_{1:k})$ by:

$$p(\boldsymbol{X}_{0:k}|\boldsymbol{Z}_{1:k}) \approx \sum_{i=1}^N \omega_k^i \delta_{\boldsymbol{X}_{0:k}^i} \tag{20}$$

where $\delta_{\boldsymbol{X}_{0:k}^i}$ is a delta function at $\boldsymbol{X}_{0:k}^i$.

At time step k, the new state $\boldsymbol{X}_k^i$ for the i-th particle is sampled from the current state $\boldsymbol{X}_{0:k-1}^i$ and all the observation $\boldsymbol{Z}_{1:k}$ according to a proposal density:

$$\boldsymbol{X}_k^i \sim q\left(\boldsymbol{X}_k|\boldsymbol{X}_{0:k-1}^i, \boldsymbol{Z}_{1:k}\right) \tag{21}$$

In other words, the new state $\boldsymbol{X}_k^i$ of the i-th particle at time step k is sampled from a distribution which takes the current state $\boldsymbol{X}_{0:k-1}^i$ and the observation $\boldsymbol{Z}_{1:k}$ as parameters.

At time step k, the weight ω_k^i is updated from ω_{k-1}^i by:

$$\omega_k^i \propto \omega_{k-1}^i \frac{p\left(\boldsymbol{Z}_k|\boldsymbol{X}_k^i\right)p\left(\boldsymbol{X}_k^i|\boldsymbol{X}_{k-1}^i\right)}{q(\boldsymbol{X}_k^i|\boldsymbol{X}_{k-1}^i, \boldsymbol{Z}_k)} \tag{22}$$

In addition, the initial state $\boldsymbol{X}_0^i$ from the joint prior distribution of the state variables, and the initial weight ω_0^i for each particle is $1/N$.

In practice, iterations of Eqs. (21) and (22) may lead to particle degeneracy problem, i.e., only a few particles have significant weights. This problem can be solved by resampling: a new set of N particles is generated from the discrete approximation shown in Eq. (20), and the weight of each new particle is set as $1/N$ again.

Some variants of the SIS algorithm have been developed in the literature to simplify its implementation, and a most widely used one is the sampling importance (SIR) algorithm [9]. The SIR algorithm takes the state transition distribution $p\left(\boldsymbol{X}_k|X_{k-1}^i\right)$ as the proposal density distribution $q\left(\boldsymbol{X}_k|\boldsymbol{X}_{0:k-1}^i, \boldsymbol{Z}_{1:k}\right)$, and conducts resampling at each iteration. So Eqs. (21) and (22) reduce to:

$$\boldsymbol{X}_k^i \sim p\left(\boldsymbol{X}_k|X_{k-1}^i\right) \tag{23}$$

$$\omega_k^i \propto p(\boldsymbol{Z}_k|\boldsymbol{X}_k^i) \tag{24}$$

It is straightforward to implement the SIR algorithm, since it only requires sampling from the distribution $p(\boldsymbol{X}_k|\boldsymbol{X}_{k-1}^i)$ and evaluating the likelihood $p(\boldsymbol{Z}_k|\boldsymbol{X}_k^i)$. Thus, this algorithm is used to monitor the asset of interest in SAFER- P^{2}IAT.

In the case where the linear-Gaussian assumption is valid, the P^{2}IAT framework also includes the ability to use the UKF to speed up the calculations. This algorithm performs both uncertainty propagation and updating analytically with relatively few samples required.

In the PF method, each particle is passed through the Bayesian Network separately to compute the outputs and likelihoods of that particle. Thus, the method is very scalable and easily run in parallel. The P^{2}IAT framework is designed to take advantage of parallel processing to speed up calculations.

4.2 Updating

Bayesian updating in the PF method is performed by likelihood-based resampling. In this method, each sample (set of values for each uncertain parameter) is tagged with its likelihood. The likelihood is the probability that this set of parameter values would produce the data that was actually collected. In the case of inspection data, the data is only detected or not-detected. The likelihood of detecting the crack is calculated directly from the POD curve and the current estimate of the crack state. Thus, particles with cracks that are consistent with the expected inspection results are retained in the update, whereas particles that are inconsistent are discarded. In the case of not-failed data, any particle that suggests certain failure would be discarded (again this data is included to correctly calculate SFPOF). Since the failure criteria is probabilistic, there is still a non-zero probability of failure.

After updating the parameter distributions, the PF method will often perform a resampling step to ensure that the particles all remain unique. The resampling step can be done with a Gaussian or Epanechnikov kernel [10]. Both of these options preserve the correlation structure and add a small amount of jitter to the particles. Adding "jitter" to a set of particles means that the value of each particle is shifted by a small random number. The shift must be small enough to preserve the distribution represented by the samples, but large enough to inject some diversity into the set of samples which potentially contains multiple identical values.

In fact, the P^{2}IAT framework is quite general and could be set up to incorporate any type of measurement as long as the value can be predicted by some kind of model. For example, if some geometric parameter is initially unknown in a wide rage, and some measurement of it is later obtained, the measurement can be used to update the parameter's distribution. By this time, it may be that the output of interest is correlated with the geometric parameter. In that case, the updating will correctly preserve the correlation.

4.3 Flight Recorder Data

The flight recorder data parameters required must be sufficient to estimate external load parameters. For the development of SAFER- P^{2}IAT, a 6DOF flight simulation software was available which computed both flight recorder data and external loading. Such a flight simulator can be used to identify which flight recorder parameters are needed to identify external load. In this case, the following nine parameters were found to be sufficient:

- Nz (vertical acceleration), Nx, and Ny
- Mach number
- Altitude
- Roll rate, roll acceleration, pitch rate, and pitch acceleration

These parameters were deemed sufficient by first consulting with experts. Second, a model was built to predict external load from these parameters. The model was able to take these nine parameters as input and predict external loads with low error across the entire operational envelope. The nine parameters are thus sufficient to predict external loads.

The flight recorder data input to P^{2}IAT must consist of these nine parameters, optionally with some missing data. Limited missing data are supported which include losing one or a few sensors for a short period of time. Long sensor outages or periods of time with no data at all will not be estimated well.

The flight recorder data input should consist of a time history of these nine parameters for each flight, along with a Flight ID column which identifies each flight. The actual value of the flight ID does not matter, but it must be unique for each flight. The time history can either be a full time history recorded at any bitrate/frequency, or can be event driven. In either case, it is important that the resolution is sufficient to capture the major load cycles of interest.

Normally, SAFER- P^{2}IAT would be run with flight recorder data taken from an actual aircraft flight recorder. However, for demonstration purposes, it can also be run on data extracted from the MES.

4.4 Mission Data

Mission data consist of a Mission ID for each flight. The mission IDs must be consistent with the missions described in the MES. For example, if the MES contains 5 missions (labelled 1, 2, 3, 4, and 5), then the mission ID for the input data can only be one of 1, 2, 3, 4, or 5 and the flight should be similar to other flights of the same mission number in the MES. The framework does not modify the description of a mission over time, and forecasting will draw from the example flights for each mission in the MES. If the definition of a mission changes (e.g. mission 2 is now flown more harshly), then the MES should be updated to reflect that.

4.5 Baseline Usage Data

The baseline usage data, also called the MES or the probabilistic load spectrum, is described in Section 3.0. It contains data for a simulated event-driven flight recorder for 5 mission types and a total of 1,000 flights. The MES contains many time hacks for each flight. Each time hack is comprised of a description, flight recorder data, a flight ID, a mission ID, and external load parameters.

4.6 Mission Model

Mission data is used to estimate the current mission mix, i.e. the probability of flying each mission. The most recent mission mix is then used when drawing random flights for forecasting. So, the framework does model changing aircraft usage & missions over time. The constraint on this model is just that the mission definition cannot change over time.

The mission mix is represented by a Dirichlet random variable with uncertain parameters [11]. To understand what a Dirichlet random variable is, consider a coin toss. The variable representing heads or tails is a Binomial random variable. The Binomial variable is parameterized by a single value, the probability of getting heads (p_H). The value p_H itself can be a random variable as well, whose distribution one might infer from some coin flip data. Now, consider the Multinomial case where a variable can take on one of say five values (rather than the two possible values in the coin flip example). The Multinomial distribution is parameterized by the probabilities of each value (p_1, p_2, p_3, p_4, p_5), where it is noted that the probabilities must sum to 1. Now, if one is to infer the values of the p_i, one must specify that the p_i must be between 0 and 1 and must sum to 1. This is exactly what is meant by saying that the p_i follow a Dirichlet distribution. A Dirichlet random variable is simply a shorthand representation of these conditions on the p_i.Thus, the parameters of the Dirichlet random variable for the mission mix are the probabilities of each mission occurring. The parameters themselves are uncertain, each having a probability distribution. The SAFER- P^2IAT framework requires the user to specify a prior guess for the values of each of the parameters (i.e. an initial guess of the mission mix). Then, as each mission ID data point is assimilated, the framework performs a Bayesian update on the uncertain parameters.

We chose to model forecasted flights using a mission mix to incorporate a moderate level of fidelity into the forecasting. A higher fidelity model might, for example, include a Markov transition model of how one mission might be followed by another. Alternatively, one could consider modeling how many times a mission is repeated before a new mission is flown. However, little data were available to investigate such patterns in missions, so it was not clear that any such added fidelity would actually be useful.

4.7 Flight Recorder Data to External Load Model

On a time-hack by time-hack basis, flight recorder data must be converted to external loads. The external loads are some parameterization of the distributed load on the asset. While this could come from a complex analysis (e.g. principal components), if the area of interest is relatively simple geometrically then a few bending moments and torques may be sufficient to define most or all loading cases. In the case of the outer wing of an aircraft, it was found that the bending and torque on the main spar at one location (i.e. two scalar numbers) were sufficient to parameterize nearly all loading conditions. These values are referred to as WBR3 and WTR3 for Wing Bending and Wing Torque, respectively, on the right side at station 3. In general, though, any number of scalar values which describe the overall external loading can be used.

The model which converts flight recorder data to loads is created using outputs from the 6DOF flight simulator. Figure 19 shows how the model is created. The flight simulator generates both flight recorder data and external loads when control input (e.g. stick input for an aircraft) are supplied. The simulator was run for a variety of maneuvers which covered the operational envelope. See Appendix A for a listing of the input conditions for the simulator. The flight recorder data and external loads were collected for all of the runs (each simulator run results in a time history with hundreds of points, each of which can be used to train the model). Then, a Gaussian Process meta-model was built to predict external loads from flight recorder data; this model was trained on some of the data from the simulator. The model was validated on the remaining data.

The model was built using the Bayesian Hybrid Model/Intelligent Design and Analysis of Computer Experiments (GE-BHM/IDACE) tool. The tool builds a Gaussian Process meta-model, training the model parameters using a Markov Chain Monte Carlo (MCMC) algorithm. The tool is an implementation of the Kennedy and O'Hagan framework [12] [13]. In IDACE mode, the tool attempts to build a simpler model by taking a subset of the data. This is done iteratively:

1. Build a model with a subset of points (training set) and predict on the rest (test set) for validation
2. Identify points in the test set with large errors; stop if error is below threshold
3. Add points with largest errors to the training set and return to step 1.

The final model predicts both a mean and standard deviation of the external load parameters for given flight recorder data.

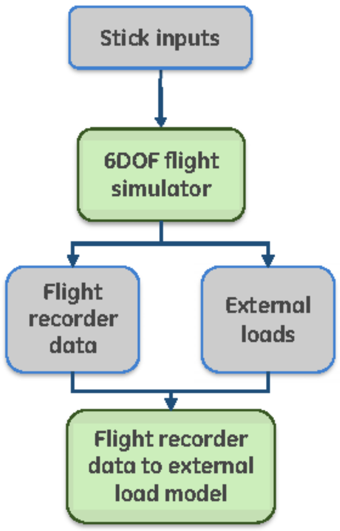


Figure 19. Process for Creating the Flight Recorder Data to Loads Model.

4.8 Usage External Loads

External loads on the structure, for example bending moments and torques, are calculated from the input flight recorder data. Since the flight recorder data is only a proxy, the loads are uncertain. In the case of a GEBHM model, the predicted external loads are normally distributed. For example, the model predicts the mean bending moment and the standard deviation of the bending moment. Since the prediction is done on a time-point by time-point basis, the external loads are an uncertain time history which retains whatever time resolution was used to capture the flight recorder data (e.g. a particular bitrate/frequency or event-driven).

4.9 Forecasted External Loads

When performing forecasting or filling in missing data, external loads are predicted using the Mission model. This model takes as input the current estimate of the mission mix and generates a random flight (load history) from a database of flights (i.e. the baseline usage data or MES). The flights in the MES were created with known external loads, so in this case the external load for a particular flight is known. The standard deviation of the external load is zero. However, there is uncertainty in which flight is chosen. In P^2IAT, a flight will be randomly drawn for each sample. The number of samples, N_{sam} is one of the main settings of P^2IAT. Note that each sample will have a different mission as well.

4.10 Geometry Parameters

Geometry parameters are inputs to the stress intensity factor model. When the stress intensity factor model is built, the training data can optionally have some geometric parameters that change. These can be geometric parameters that influence the stresses (e.g. size/shape of parts) or parameters which influence the Green's function (e.g. hole diameter, edge distance), or both.

In choosing which geometric parameters to vary, the approach is generally to first consider some candidate parameters based on expert knowledge, engineering judgement, and how difficult it is to modify those parameters in a stress model (e.g. mesh morphing in a finite element model) in an automated (i.e. scripted) way. Automation is necessary because building the stress intensity factor model will require multiple runs of the stress model in different geometrical and load conditions. One must then perform some initial trails to assess sensitivity and down-select the parameters. The trials can be one-dimensional parametric studies or full multi-parametric studies. In the case of a full study, one can build a GEBHM meta-model and use the automatically generated sensitivity information to decide which geometric parameters are most important.

Once the parameters have been selected, realistic ranges and probability distributions for the parameters should be estimated by examining engineering data and consulting expert judgement. For example, for bolt hole diameter, it may be uncertain whether a fastener is the design size or an oversize fastener. For a bolt hole offset, one can estimate a nominal maximum offset based on the manufacturing method (whether hand-drilled, or milled).

For reference, the following geometric parameters were chosen for the single point demo:

- Thickness: fixed to nominal
- Bolt hole diameter
- Bolt hole offset
- Edge distance

Geometrical uncertainty is then propagated through the P^2IAT process by setting probability distributions for these parameters.

4.11 Initial Crack State

The initial crack state in the flow chart of Figure 17 refers to first of all the very initial crack state for the material of interest (as-manufactured). This is characterized as the equivalent initial flaw size distribution (EIFS).

The initial crack state may also refer to some known initial state from, e.g., a previous run of P^2IAT. This will often be the case when usage data is fed to the P^2IAT system in periodic interval. When usage data is supplied, the damage is grown for that period of time and the result is a new estimate of the crack state at the end of the supplied usage data. This final crack state is then used to initialize the next run of P^2IAT, when the next batch of usage data is supplied.

The initial flaw size used in any Paris equation-type fatigue crack growth calculations is critical for realistic life calculations. Crack length measurements, with associated usage data, would be difficult or impossible to procure for this program. Thus, whatever data might be available in the public literature was used to inform the equivalent initial flaw size distribution.

The most appropriate source of data for initial flaw sizes is S-N data. There is abundant data in this format available, for various materials, coatings, types of holes, etc. S-N data also represent fatigue crack growth, mostly in the small crack regime. In contrast, da/dN data is not appropriate because it is usually gathered from pre-cracked specimens, so the initial/small crack is not well represented. Thus, S-N data is a good source for informing the initial flaw size.

However, S-N data is traditionally only used to fit a linear damage model, called Miner's rule. In this model, the damage grows from 0 to 1 over the N cycles that the specimen took to fail. The P^2IAT framework, however, has a non-linear crack growth model, the Walkerized Paris equation model. The model describes the crack growth rate as $\frac{da}{dN} = \frac{C_0}{(1-R)^{m(1-\gamma)}} \Delta K^m$, where R and ΔK describe the applied load. The model of course requires a starting crack size a_0 as well. Thus, the overall damage model has four parameters - C, m, γ, a_0. It would at first seem difficult or impossible to fit all four parameters with a single point of S-N data. Indeed, with only S-N data, one could just as well assign a large initial flaw size with a slow crack growth rate (C) as a small initial flaw size with a faster crack growth rate. Knowing only the endpoint, the number of cycles to failure, one cannot distinguish between these two possibilities.

The comparison of Miner's rule and Paris equation crack growth are shown in Figure 20. While it is rare to see both models plotted together, it does highlight the fact that they are two models of the same phenomenon. Miner's rule is formulated as non-dimensional damage, while Paris equation predicts crack length directly. In order to compare them, one can consider that damage equals 1 corresponds to failure. For this report, failure of a specimen can be related to crack length by assuming that when the crack length reaches the width of the specimen, the specimen has failed. One could also use a fracture toughness criteria for failure; the results look almost identical for the example below. Thus, Miner's rule and Paris equation crack growth can be directly compared in terms of crack length over time.

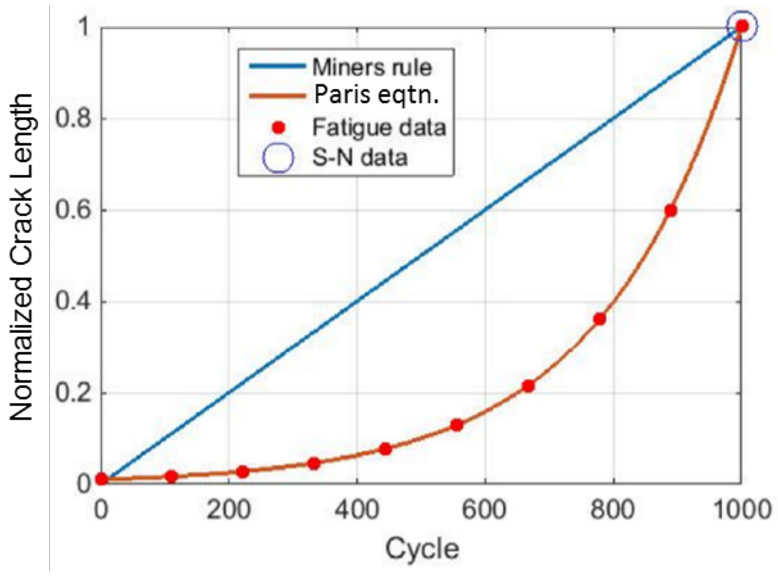


Figure 20. Comparison of Miner's Rule and Paris Equation Crack Growth Models.

Figure 21 shows the high density of da/dN data which allows for fitting of crack growth rate parameters. Again, though, S-N data is only given at the end point, making it difficult to fit a full Paris equation model with all parameters. The remedy is simply to use known values of $C, m,$ and γ from da/dN datasets, and then only fit a_0 with the S-N data. That is, given a known crack growth rate, one can find the initial flaw size that would produce a large crack (failure) after some known number of cycles N. Other researchers have also found ways to fuse da/dN and S-N data [14].

The methodology used is the basic data-fitting tool BHM in the pure MCMC mode. For demonstration, the values of C and m are fixed. The data are input as a stress level $\Delta\sigma$ and the number of cycles to failure N, along with the specimen width. Then, the BHM model is built by finding those values of a_0 which, when put through Paris equation crack growth and the constant amplitude stress value, result in failure at N cycles. The result is a distribution of a_0, the EIFS distribution, which is centered around the most likely values (those that best reproduce the S-N data). Of course, the width of the distribution is representative of the spread in the experimental data, which is often significant. Finally, note that for robustness the statistical comparisons are done on $\log N$ (as opposed to N or the crack length).

To find the EIFS distribution for 2124-T851, as this is the material at the demonstration location, we use data from a technical report which contains both da/dN and S-N data for the same material and laboratory [15]. Figure 21 shows the da/dN data and the Paris equation fit (linear in log-space). Note that C is the intercept and m is the slope of the solid line. Figure 22 shows the S-N data used to calibrate the initial flaw size. The upper right diagram of the test specimen shows that the relevant maximum length is 0.3 inches, so failure is considered to occur when the crack reaches this length.

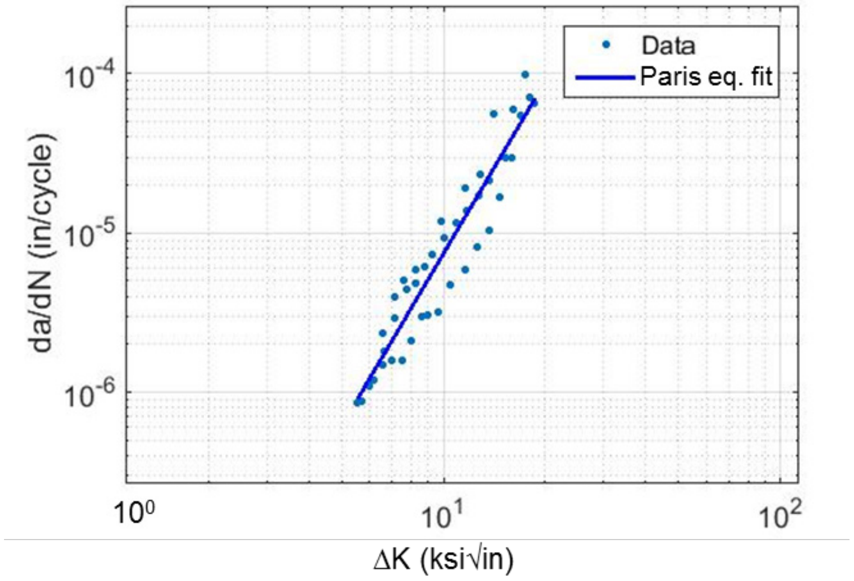


Figure 21. Fatigue Crack Growth Rate Data for Aluminum 2124-T851.

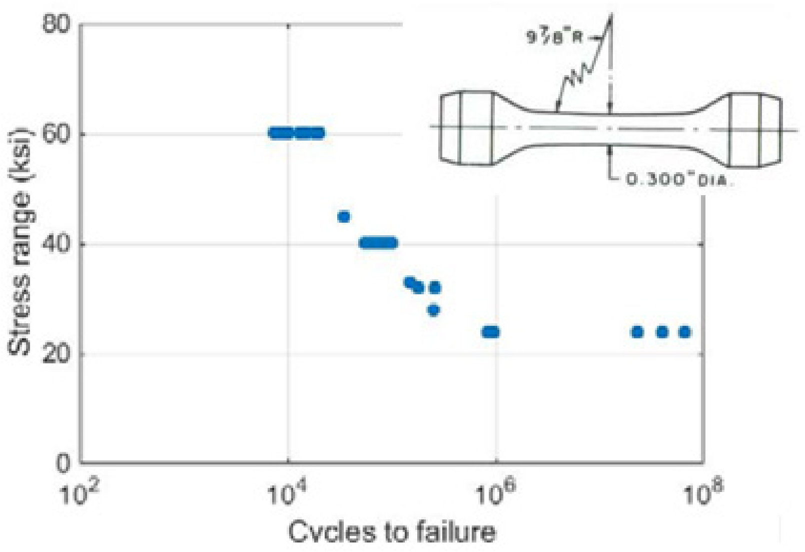


Figure 22. Stress-Life Data for Aluminum 2124-T851.

Figure 23 shows the resulting calibrated EIFS distribution. The distribution is approximately log-normal, with a mean of about 0.00023 inches. This is on the low end of what is expected, but it is not unreasonable. Expert knowledge suggests that the EIFS should be somewhere between 0.0005 inches and 0.050 inches. The most important result of Figure 23, though, is that the spread in the EIFS distribution is only about 1 order of magnitude, from 0.0001 to 0.001 inches. Since the initial flaw size has such a strong impact on life predictions, it is important to characterize it as accurately as possible, which in this case means having as narrow a distribution as possible.

Figure 24 shows what the calibrated EIFS distribution would predict for the S-N coupon tests. The horizontal axis is the actual number of cycles to failure and the vertical axis is the number of cycles to failure predicted by the Walkerized Paris law model using the calibrated EIFS distribution. The plot shows that the model is reasonably accurate especially for moderate stress levels.

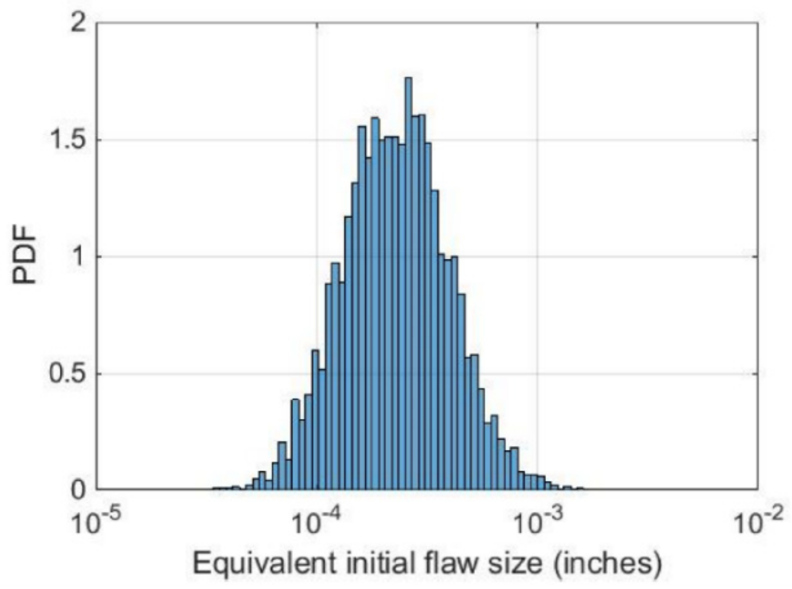


Figure 23. Calibrated Equivalent Initial Flaw Size Distribution from S-N Data

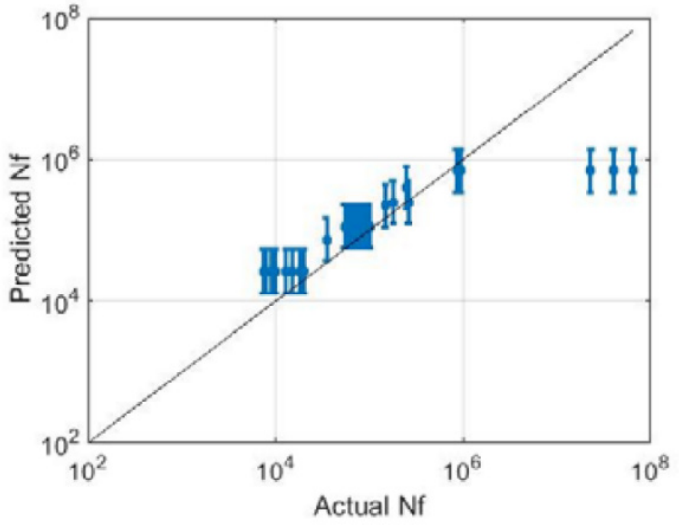


Figure 24. Prediction Errors of the Paris Equation Model at the S-N Data Points

4.12 Stress Intensity Factor Model

A stress intensity factor model, built for each control point of interest on an asset, is a computational model that takes external load, geometry, and crack state parameters as input and produces stress intensity factors as output. The stress intensity factor is the driving force which opens the crack at the control point, and in general it can be computed using the Green's function approach (Note, the current work only considers mode I stress intensity factor, crack opening mode, although other modes could be included as well):

$$K_I = \iint_{crack\ face} \sigma_\perp(x,y) \cdot G(x,y)\, dx\, dy \tag{3}$$

where $\sigma_\perp(x,y)$ refers to the stress component perpendicular to the crack plane (i.e. the crack opening stress) and $G(x,y)$ is the Green's function. Stress is generally calculated from a finite-element model, while the Green's function is usually assumed to take a particular form for a given crack type. A typical deterministic analysis would calculate a single value of K_I for a nominal geometry and a given crack state and loading condition. For P^{2}IAT, though, it is necessary to be able to calculate K_I on demand given all of the parameters that are not known apriori. The parameters of the K_I model must include the external load parameters (e.g. bending moment and torque) and the crack state. Any additional parameters can be included to allow P^{2}IAT to propagate their uncertainty.

For demonstrating P^{2}IAT, the K_I model has two external load parameters, four geometrical parameters, and two crack state parameters. Several assumptions have gone into building the model which will be listed below. The analysis steps to build the model are shown in Figure 25. This is of course not the only possible way to build a stress intensity factor model, but it is the approach used in the demonstration of P^{2}IAT. The reasons this approach was taken are as follows:

- To demonstrate the ability to incorporate both a coarse global FE model and a fine local FE model
- To demonstrate the ability to account for geometrical uncertainty by morphing the local FE model geometry
- To use a moderate resolution stress intensity factor model which considered the entire stress field on the crack plane from the local FE model
- To reduce analysis requirements by not accounting for multiple crack planes, curving crack trajectories, or how the presence of a crack modifies the load paths in the structure. Such analyses could be included by extending the flow chart (e.g. an arrow from "crack state" to "local FEM" to signify that the local FE could be modified to include the crack itself)

Once the crack opening stress model and the Green's function model have been developed, they can be combined relatively easily to produce the stress intensity factor model. This is done by defining a set of training points (i.e. a DOE or design of experiments) where each of the inputs are varied (some inputs, such as bolt hole diameter, may be inputs to both models). Then, for each set of inputs, the stress field and Green's function are calculated over the crack face. A simple 2D integration scheme can be used to calculate K_I. The result is a table of inputs and a K_I value for each. Finally, a GEBHM model is built to capture the overall input-output relationship:

$$K_I = K_I(load, geometry, crack\ state) \tag{4}$$

Another important factor is the distinction between corner and through cracks for bolt holes. Generally, a crack will start out as a corner crack and transition to a through crack partway through the crack growth. In P^{2}IAT, separate corner crack and through crack K_I models are built. The corner crack model is a function of both the crack length (along the surface) and depth (into the thickness), usually denoted c and a, respectively. There are two separate Green's functions for a corner crack, one which represents growth in the surface direction and one in the depth direction. Thus, the corner crack K_I has two outputs: $K_{I,depth}$ and $K_{I,surf}$. The crack is assumed quarter-elliptical and P^{2}IAT tracks c and a while the crack is a corner crack. Once the crack transitions to a through crack, a different Green's function is used and only the surface crack length is retained.

P^{2}IAT can handle residual stress effects by building yet another stress intensity factor model. The stress intensity factor due to external load can be linearly added to that due to residual stress. In this way, uncertainty due to how fasteners are installed and the effects of Taper-Lok fasteners can be incorporated into P^{2}IAT.

The following paragraphs describe in more detail the analyses in Figure 25 and how the stress intensity factor models were built for demonstrating P^{2}IAT.

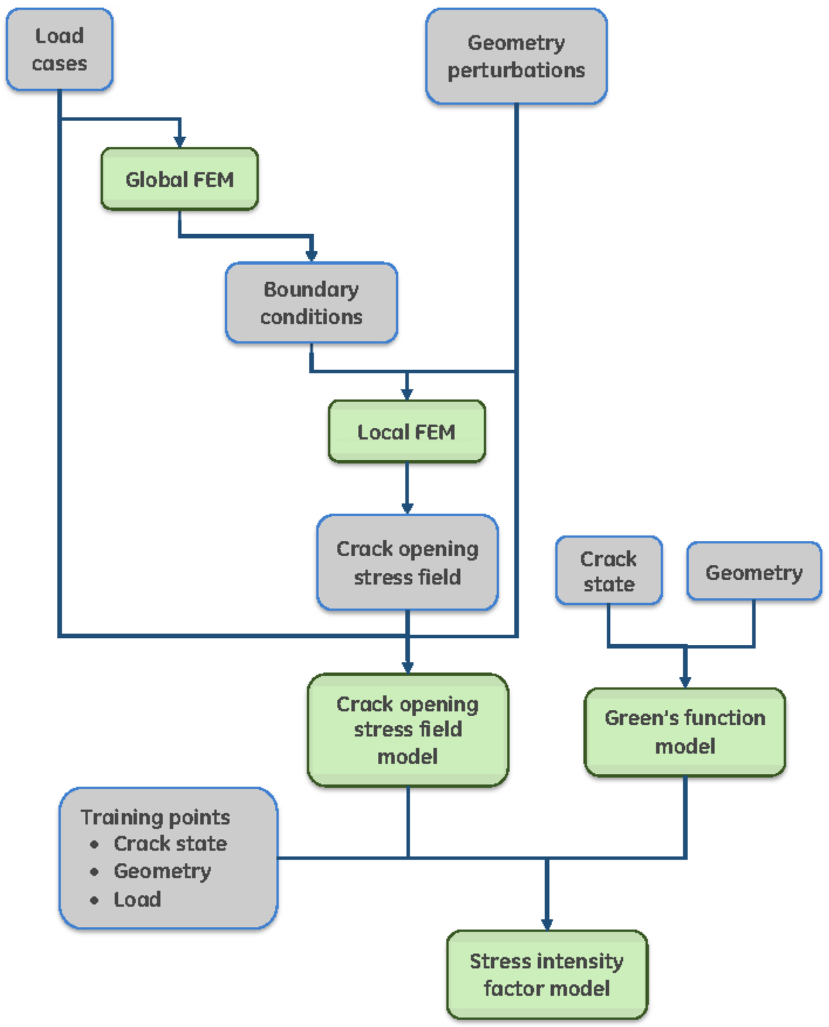


Figure 25. Process for Creating Stress Intensity Factor Model.

4.12.1 Global and local FE models

A local finite element model is used to evaluate local stress fields where cracks are expected to form. Currently, a local model for the demonstration control point has been built. Once built, the stress field can be evaluated for a variety of load cases. However, an additional source of uncertainty is the geometry. That is, the hole locations, diameters, etc. are not known precisely. By morphing the mesh, this uncertainty can be quantified. The overall strategy is to solve the finite element model for various load cases and geometries (i.e. "samples"). Then, a BHM model is built to relate the local stress field to the input variables. If the model accuracy is low, new samples which will improve accuracy are identified using IDACE. These new FE runs are performed and the model re-built. The process is repeated until sufficient accuracy is achieved.

The overall analysis approach is as follows:

1. For a given load case, run the global (air vehicle) FEM to generate boundary conditions for the local FE model

2. For given geometry parameters, morph the local FE mesh
3. Apply the boundary conditions to the morphed mesh and solve
4. Identify the location of maximum stress at the area of interest (e.g. extending from the inner wall of a bolt hole into the bulk material) and extract the stress on the assumed crack plane.

Once these steps have been performed for a series of load conditions and geometry parameters, a BHM model can be built that predicts the stress field on the crack plane as a function of the load and geometry. This model was then used to evaluate the stress intensity factor.

After the BHM model is built, input points that result in large uncertainty are identified and additional load/geometry cases run. The BHM model is updated when these runs are finished, and the process continues until a proscribed error tolerance is reached.

For the demonstration, the load cases are derived from the 6DOF flight simulator runs. However, some of the load cases generated by the simulator are not realistic. Each load case must be checked to ensure that the simulator is producing consistent, realistic load cases. Twelve load cases were selected by hand to be well-distributed across the input space of external loads (bending and torsion at the outboard wing station). It was also verified that the load cases were well distributed across the root bending, torsion, and shear space. The cases were chosen from the library of maneuvers run through the flight simulator. Two load cases were found to be invalid. In particular, they came from 180 degree rolls and were load cases that occurred while the aircraft was inverted. These two load cases were disregarded from further analysis. It is possible that the resulting stress intensity factor model will have large predicted uncertainty in the area of the input space covered by these two load cases; if so, then it may be possible to select other, valid load cases to include in the model building. In the end, this was not found to be necessary for the P^{2}IAT demonstration.

Once boundary conditions for the local FE were extracted for each of the twelve load cases, a DOE was generated for all of the input variables (load and geometry). The geometry variables had to be selected in a special way to minimize mesh distortion caused by the mesh-morphing procedure. Large changes in bolt hole location or diameter, or both, can cause mesh distortion so great that a solution could not be reached. Thus, a two-stage DOE was developed. First, a set of 16 points were generated with relatively small geometry perturbations. These points were distributed in a Latin hypercube [16]. Then, an additional 4 points were chosen which had larger variation but were restricted to vary only the diameter or the hole position, but not both. This allowed some assessment of larger variation while not creating too much mesh distortion.

Once the finite element models were run, post-processing was done to extract the stress on the anticipated crack plane.

The demonstration control point consists of five bolt holes. In using the local FE for crack growth analysis, a decision must be made as to how the five holes will be analyzed – will each hole be treated separately? Or does only the highest stressed hole need to be analyzed? Or can there be some other way to combine the holes? The hole that was chosen had cracking in previous full-scale fatigue tests of these types of wings. In general, though, one would normally consider the hole that generally has the highest predicted stress across the FE runs. In the case where some load conditions cause one bolt hole to have high stress, while other conditions result in highest stress at a different hole, it can be difficult to predict where the crack will actually form. In this case, previous experience and engineering judgement should be used, while noting that one may consider multiple locations as separate control points.

In addition, the assumption of a single planar crack must be made carefully. While a crack may begin at the location of maximum tangential stress, the crack path is not necessarily straight. Still, for computational simplicity, we would like to extract a single crack plane from the local FE model. In addition, the location and orientation of the maximum stress changes as the load and geometry parameters change. That is, the location and orientation of the crack plane changes with load parameters, geometry parameters, and of course time. An assumption must be made in order to compare all of the local FE results and create a BHM model to predict the generic crack plane stress.

In this work, we assume that while a single crack will eventually form, at any time the relevant stress plane includes the maximum principle stress at that loading condition. That is, we take the conservative approach of taking the plane of the maximum principle stress, even though this may not be where the eventual crack forms. In addition, we assume of a planar crack and use the stress normal to this plane which may be inaccurate and potentially non-conservative. Compared to the other sources of uncertainty in the overall P^{2}IAT approach (e.g. using 1D da/dN crack growth curves), though, the inaccuracies are assumed to be small.

Examples of the local FE model, a few candidate bolt holes, and mesh-morphing are shown in Figure 26. The upper left picture is the entire local FE without the front spar showing the five fastener holes in yellow. The upper right picture shows the three holes that will be morphed. The bottom picture shows the initial meshes in green and the morphed meshes in red. Morphing modifies the positions and diameters of the holes. Scripts were developed to perform the morphing given a new hole position and diameter, solve the FE models, and extract the crack opening stress field.

The load cases chosen for the local FE model runs are shown in Figure 27 in the outboard bending-torque space. Each load case had to be verified to have accurate distributed loads. The load cases identified as invalid are shown in red in Figure 27. These two load cases were removed from the study. Note also that practical considerations resulted in torque limits for the demonstration, also shown in Figure 27.

Figure 28 shows the locations of the assumed crack plane for all of the local FE runs and each of the holes. Clearly the crack plane changes with the load case. The conservative approach, though, is to collapse all of the planes and assume that the crack forms with the highest stress (rather than the stress at a single location). A less conservative approach would be to estimate a fixed location for the crack plane and evaluate the stress at that plane for all load cases. Since this would only coincide with the maximum stress in a few cases, most of the time the stresses at this plane would be lower than the maximum stress. Therefore this approach would have lower stresses compared to taking the maximum stress for each load case. Of course, in reality the crack does not form on a single plane. Often a number of small cracks will form at different locations and will eventually coalesce into one major crack. Without attempting to model the coalescence process, the conservative approach can be thought of as jumping to the fastest growing crack at every applied load condition.

In Figure 28, the horizontal and vertical axes are the global wing coordinate systems. The small red circles indicate the center of the bolt holes and the blue circles indicate the edge of the bolt hole. The area outside of the blue circles represents the wings skin material. The blue lines radiating from the bolt holes are the crack planes, in the sense that the maximum stress is normal to these planes. The crack planes extend into the wing skin material about 0.20 inches which is far enough that the stress relaxes to a constant far-field value. The color of the crack plane lines indicates the stress level. Note that each crack plane is representative of a different load case, so the position of the crack planes are not necessarily correlated to the stress levels in any way. Rather, the plot simply reflects the set of load cases that were analyzed.

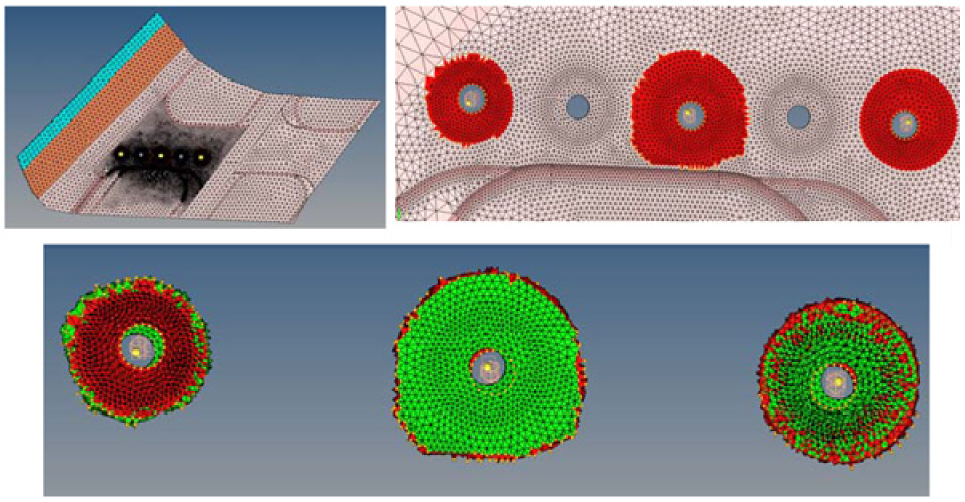


Figure 26. Images from the Local FE Model of the Demonstration Location.

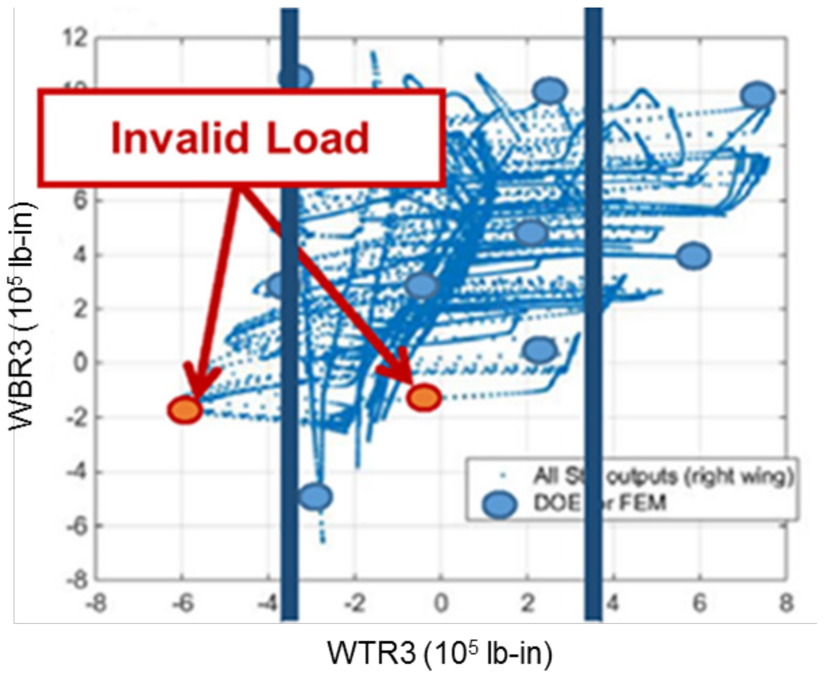
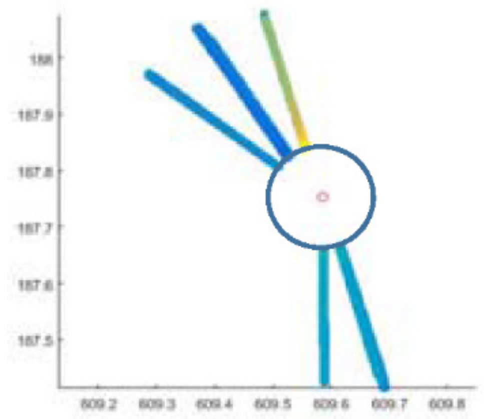
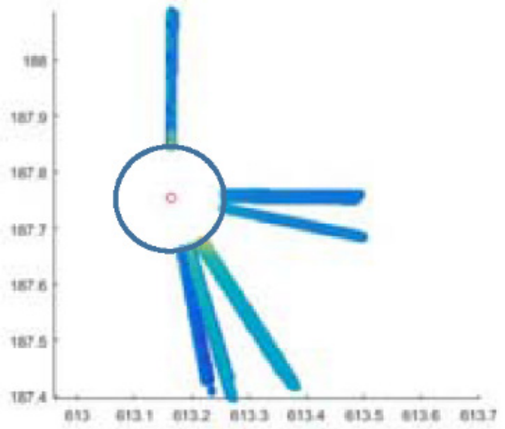


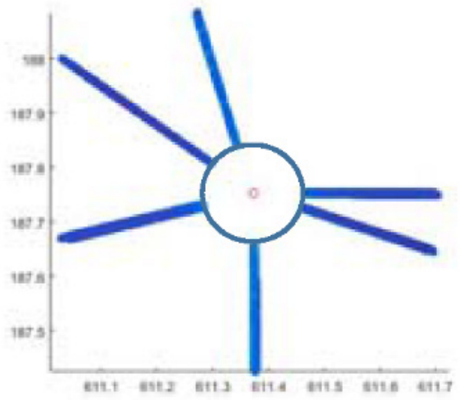
Figure 27. Load Cases Selected for the Local FE Model Showing Invalid Load Cases and Torque Limits.



a) Crack Planes for Hole 0



b) Crack Planes for Hole 1



c) Crack Planes for Hole 2

Figure 28. Crack Planes around Bolt Holes for Various Geometry and Load Case Parameters.
The Red Circle Indicates the Center of the Hole.

4.12.2 Green's Function

The Green's function has a relatively simple form for through the thickness cracks. For quarter-elliptical corner cracks at bolt holes, however, analytic forms are not available. Lockheed Martin has run a number of very fine grid finite element simulations with cracks that can be interrogated to get pointwise values of G. Many FE runs were done in which the load point, evaluation point, and geometry were varied. The load point is parameterized by the radial and azimuthal location, (r, θ). Two evaluation points were chosen, one near the surface and one into the depth of the material (parameterized by θ_e). This reflects the quarter-elliptical crack shape which is defined by a length along the surface (c) and along the thickness (a). The geometry is parameterized by the crack aspect ratio a/c and c/R, where R is the radius of the hole. The geometry is described in Figure 29.

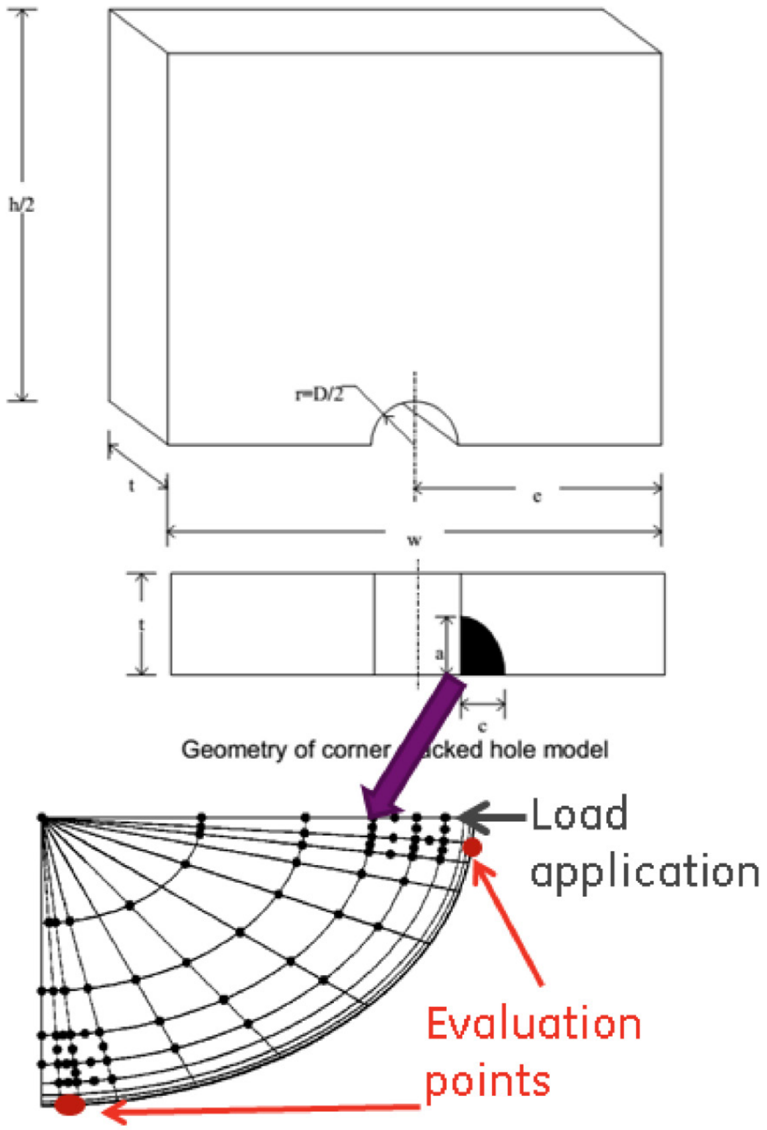


Figure 29. FE Model Definition for Green's Function [17].

To get a predictive model of the Green's function at any point, the database of fine grid FE models run by Lockheed Martin was used to train an Artificial Neural Network (ANN). The resulting neural network is a predictive model that interpolates between the data points from the database. It can predict the Green's function value at any point on the crack face given geometry parameters (e.g. a/c and c/R).

In addition to the corner crack stress intensity factor work described above, it is also necessary to build K_I models for through-cracks. The transition from corner to through crack will occur when the crack length reaches the thickness of the material. In P^{2}IAT, this transition will occur separately for each sample. Thus, at a particular time point, some samples may be using corner crack models while some may be using through crack models, since each sample has a different crack state.

The difference between a corner and through crack comes purely from the Green's function – the stress remains the same. A through-crack, 1D Green's function was derived from the Tada K-solution [18]:

$$G(a,b,y) = \left[\frac{2}{\sqrt{\pi a}(1-x)^{\frac{3}{2}}}\right]\left[\frac{g_4(x)y^3+g_3(x)y^2+g_2(x)y+g_1(x)}{\sqrt{1-y^2}}\right] \tag{5}$$

Here, y is the distance from the hole boundary to the point at which the Green's function is evaluated. The crack dimension is just the length along the surface, a (note that $0 \le y \le a$). The distance from the hole boundary to the edge of the plate is b and x is defined as $x = a/b$. The functions g_1 through g_4 are defined as follows:

$$g_1 = 0.46 + 3.06x + 0.84(1-x)^5 + 0.66x^2(1-x)^2 \tag{6}$$

$$g_2 = -3.52x^2 \tag{7}$$

$$g_3 = 6.17 - 28.22x + 34.54x^2 - 14.39x^3 - (1-x)^{\frac{3}{2}} \\ -5.88(1-x)^5 - 2.64x^2(1-x)^2 \tag{8}$$

$$g_4 = -6.63 + 25.16x - 31.04x^2 + 14.41x^3 + 2(1-x)^{\frac{3}{2}} \\ +5.04(1-x)^5 + 1.98x^2(1-x)^2 \tag{9}$$

Because the stress varies with the depth, the stress and Green's function are integrated over the crack length (surface) at various depth locations. These are averaged across the thickness to get an overall stress intensity factor of $K_{I,thru}$. Once a DOE of $K_{I,thru}$ values are calculated for the various parameters (loads, geometry, etc.), an overall BHM model for $K_{I,thru}$ as a function of load, geometry, and crack state parameters is built.

The final K_I models use the stress at the bottom corner of the assumed crack plane in the skin for the demonstration location, even though cracks found in the damage tolerance assessment were on the top corner. Using the stress at the bottom corner gives physically realistic values and is consistent with the Finite Element model predictions that the bottom surface of the skin has the highest stress.

4.12.3 Residual Stress

There are many possible approaches for quantifying the effect of residual stresses on crack growth behavior, from simple knockdown factors to 3D finite element simulations. After weighing the advantages and disadvantages of various methods, the team has decided to proceed with a 1D residual stress field superimposed on the FE stresses due to its ease of implementation and moderate accuracy level. Analytical residual stress fields developed by Lockheed Martin were leveraged [17]. Using the linear elastic assumptions of the finite element model, one can develop separate stress intensity factors for external load and residual stress that can be summed.

The residual stress field calculation assumes an isotropic expansion of a bolt hole in either thin (plane-stress) or thick (plane-strain) metal during cold-working or installing an interference fit fastener. The analytic method is based on the expansion of a thick-walled cylinder with an outer radius equal to the edge distance. The solution for a given interference level is obtained by a simple 1D root-finding algorithm to solve the complex analytical equation. The result is the full residual stress field near the hole of interest, from which the crack-opening (or closing) stress is taken.

The residual stress predictions are one-dimensional and should be the same at all azimuthal locations. The stress is calculated using either a plane-stress (thin plate) or plane-strain (thick plate) assumption. Variation in the depth direction can be considered by interpolating between plane strain and plane stress. In the case where the skin thickness is small, the entire thickness can be regarded as plane stress. The ASTM guideline for guaranteeing that plane strain conditions exist is [19]

$$t > 2.5 \left(\frac{K_{Ic}}{\sigma_y}\right)^2 \tag{10}$$

In the case where plane stress can be assumed, the residual stress is constant in the thickness direction as well as the azimuthal.

Once the residual stress field is computed, it can be integrated with the Green's function to get the stress intensity factor for corner and through cracks. Again, a BHM model is built to predict residual-stress based stress intensity factor as a function of the crack state, cold-working and fastener parameters, and other uncertain parameters representing model uncertainty (e.g. strain hardening behavior which must be assumed in the residual stress calculations). The model is built in the same fashion as described in Figure 25. Initially, many inputs to the residual stress calculations were considered uncertain to assess their effects. Many parameters had little effect within the range of possible values suggested by literature. The final set of important uncertain parameters is:

- I_{0f}, the diametral interference level (percentage) for the fastener interference
- I_{0c}, the diametral interference level for the cold working

- R_b, the distance from the center of the bolt hole to the location where the stress drops significantly
- R, the hole radius

Ranges for interference levels should be obtained from standard practices for the manufacturing process (e.g. the cold-working tool vendor's specifications or the interference fit fastener vendor's specifications). For training the model, interference levels from zero to 10% were used. Also, note that the edge distance and hole radius also affect the Green's function.

The training prediction plot for BHM model of residual stress-induced stress intensity factor in corner cracks ($K_{I,res}$) is shown in Figure 30. As expected, there are both large positive and negative values of $K_{I,res}$. A BHM model for through-cracks was built as well, with training and sensitivity plots shown in Figure 31 and Figure 32.

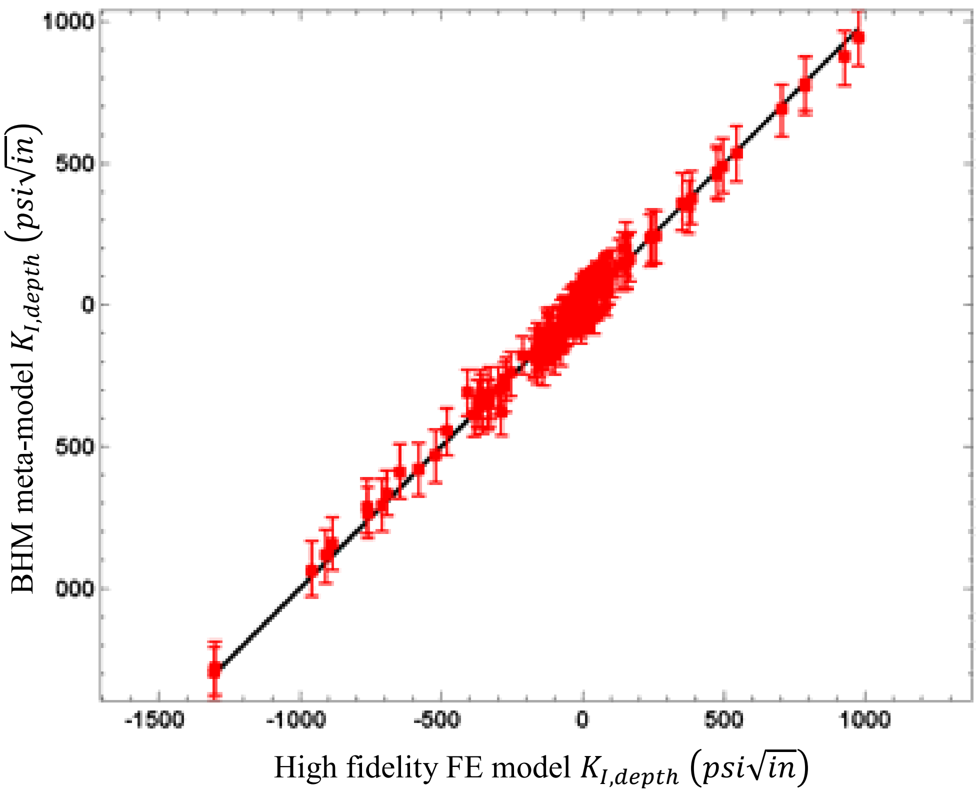


Figure 30. Training Predictions for Residual K_I Model.

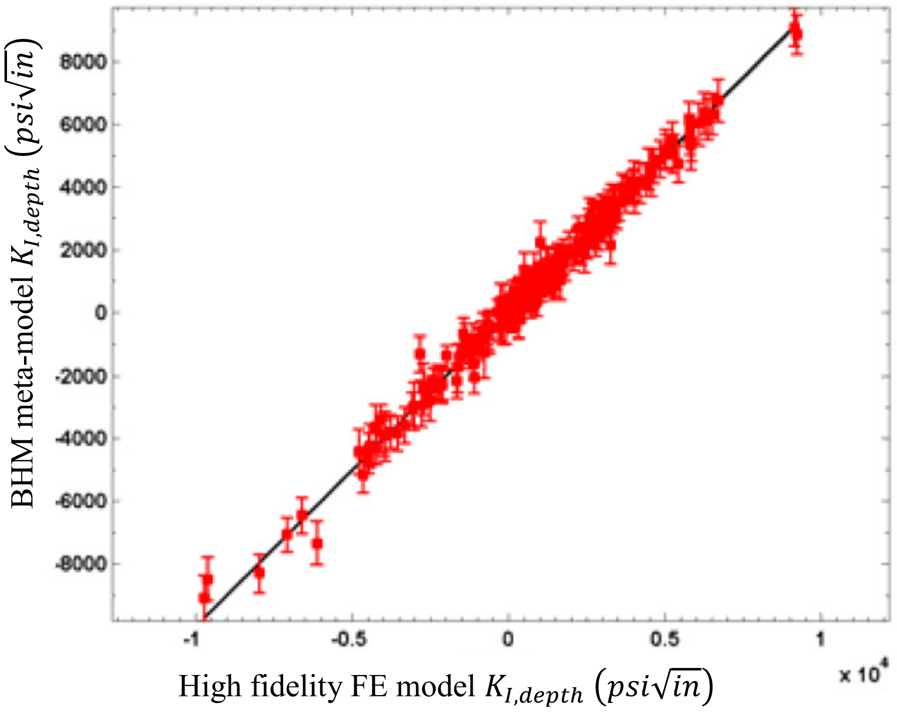


Figure 31. Residual Through-Crack Stress Intensity Factor Model Training Predictions

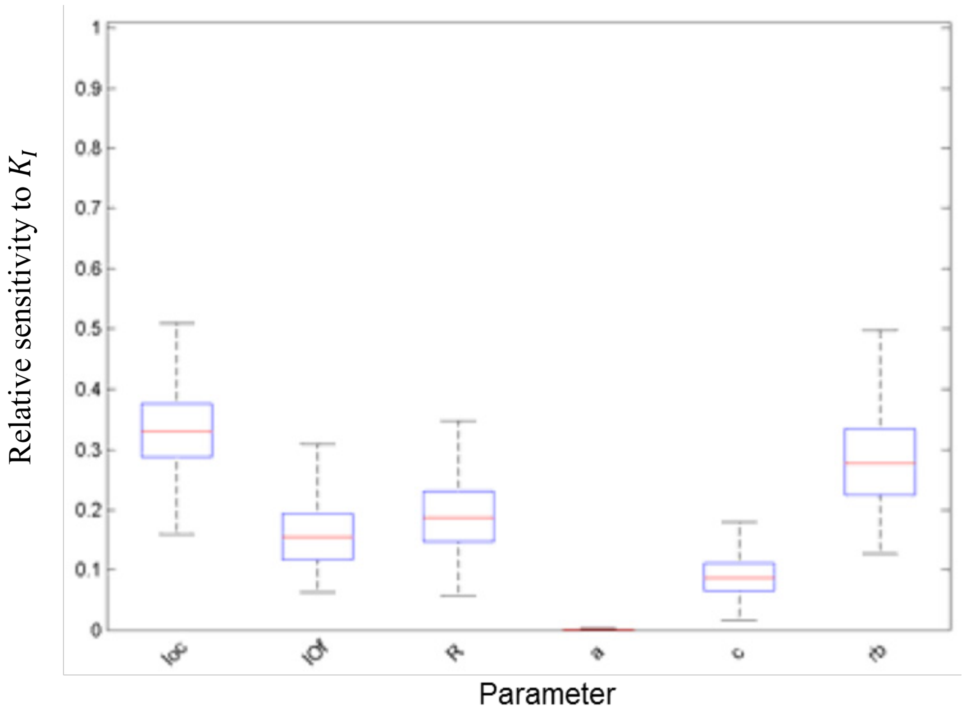


Figure 32. Residual Through-Crack Stress Intensity Factor Model Sensitivity

4.13 Crack Growth Rate Parameters

Fatigue crack growth is modeled with a Walker model in P^{2}IAT:

$$\frac{da}{dN} = \frac{C_0}{(1-R)^{m(1-\gamma)}} \Delta K^m \tag{11}$$

where R is the stress ratio of a load cycle and ΔK is the stress intensity factor amplitude of the load cycle. The Walker model can be fit to da/dN test data where it is reported as a function of ΔK and a particular stress ratio. Data was extracted from the Damage Tolerance Design Handbook [20], digitized using the Grabit Matlab tool [21]. A 2D linear least-squares fit was performed to get the coefficients. The fit is done in the log space:

$$\log(da/dN) = m\log(\Delta K) - m(1-\gamma)\log\big(1\ - R\big) + \log(C_0) + \epsilon_{MAT} \tag{12}$$

The fit returns two probabilistic features which are retained and used in P^{2}IAT:

- The covariance matrix of the coefficients
- The standard deviation of the residuals (i.e. model error, ϵ_{MAT})

In P^{2}IAT, the coefficients are treated as correlated normally distributed inputs and the model error is an additional independent zero-mean normally distributed input. Note that the covariance matrix is only valid when one uses the coefficients of the fit directly. In the Walker model, the coefficients are m, $(m(1-\gamma))$, and $\log(C_0)$, so these are the variables that are specified as correlated normal (not, e.g., γ or C_0). The code itself must transform the coefficients back to useful variables like γ and C_0.

Uncertainty in fatigue crack growth parameters represents material property scatter. Thus, it stands to reason that with enough data, one could learn the material properties of a given component by using the P^{2}IAT framework and supplying sufficient data for updating. Indeed, this is certainly possible. From the team's experience, though, the uncertainty in initial flaw size and loads tend to outweigh material property uncertainty. Thus, in order to learn the fatigue crack growth parameters, one would require significant knowledge of the initial state and loading of the asset in question.

4.14 Fatigue Crack Growth Model

The fatigue crack growth model performs cycle-by-cycle crack growth. The input stress intensity factor history is cycle-counted using a rainflow algorithm. Note that this results in stress intensity factor cycles directly (ΔK), without any intermediate load or stress cycles. The Walker model (see Section 4.13 Crack Growth Rate Parameters) calculates the crack growth rate from the cycles which is modified by a load interaction model. The crack length is incrementally increased for each cycle. The load interaction model also keeps track of the plastic zone size near the crack tip induced by overloads. The model is applied separately to the surface and depth directions of crack growth in a corner-crack. The crack growth model is implemented in code directly. One could build yet another BHM model for this, but it would be less accurate and it would be difficult to specify the load history in a flexible way. It can be done by assuming particular load histories, which is essentially what is done in forecasting anyway.

4.15 Rainflow Cycle Counting

The rainflow counting method is one of the most widely utilized of the several methods compiled in [22] and it is schematically described in Figure 33.

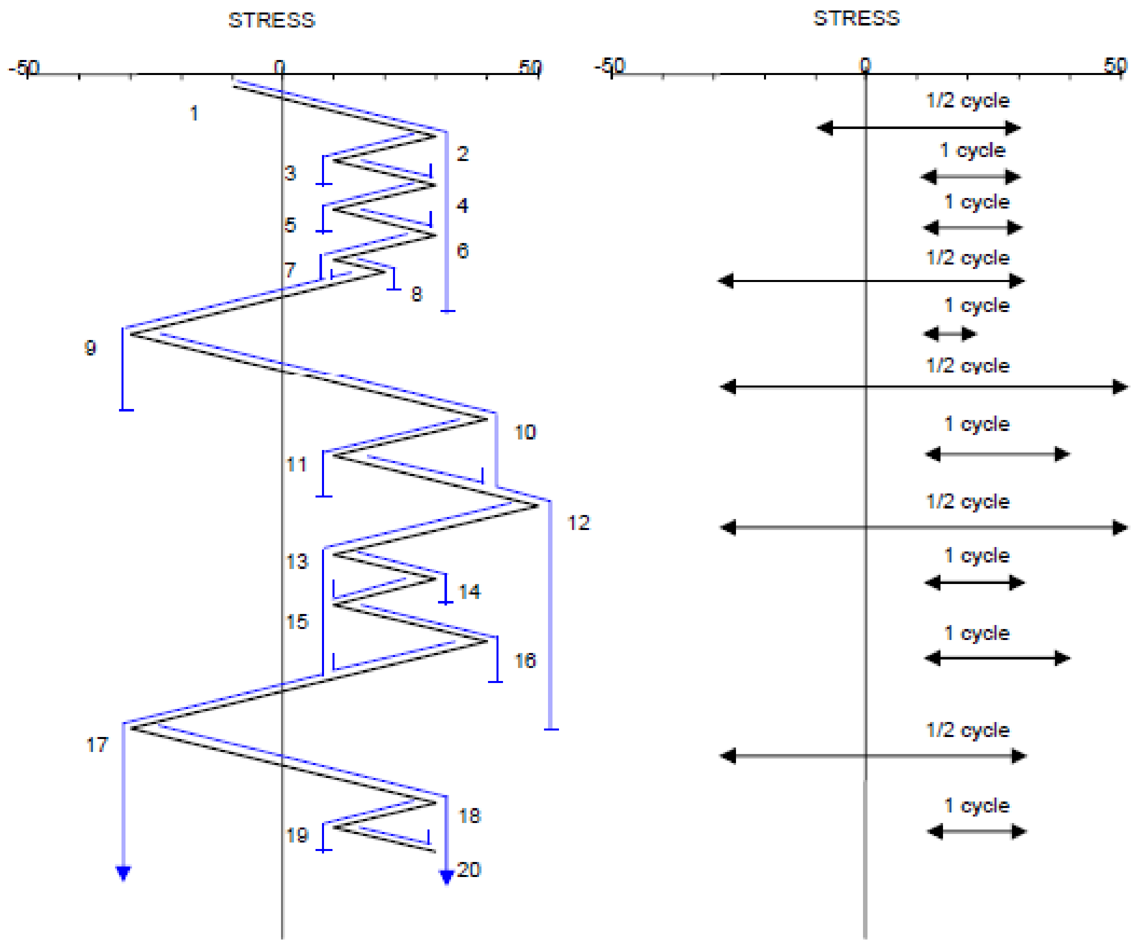


Figure 33. Schematic Representation of the Rainflow Methodology for Cycle Counting.

One particular outcome from the standard rainflow counting method shown in the right-hand side of Figure 33, is a half-cycle which is also called as a reversal. Although there is no consensus on how to treat the half-cycles, one approach is to use the simplified rainflow method for repeated loading. The name comes from the fact that load events are repeated to close the otherwise incomplete hysteresis cycles, such that only full-cycles are accounted for.

The python implementation of simplified rainflow counting was verified to produce the same results as the Lockheed Martin standard code. Details are provided in Appendix B.

Also, note that load cycles with negative stress ratio (R) are usually treated differently from those with positive values. In this work, the simple approach of taking $\gamma = 0$ for cycles with negative stress ratio was used.

4.16 Load Interaction (Retardation) Model

The Wheeler retardation model attempts to capture how overloads produce a region of strain-hardened metal near a crack tip [23]. The crack will tend to grow more slowly through the plastically deformed material. Thus, moderate load cycles following a large overload will have retarded crack growth rates.

The Wheeler model used in this work is formulated as a multiplicative factor on the crack growth rate:

$$\frac{da}{dN} = \phi \cdot \left(\frac{da}{dN}\right)_{DTDH}, \quad \phi = \begin{cases} 1, & a + r_{pi} \geq z \\ \left[\frac{r_{pi}}{z-a}\right]^{r_e}, & a + r_{pi} < z \end{cases} \tag{13}$$

where the radius of the plastic zone (assumed circular) induced by the current load is termed r_{pi} and is calculated as:

$$r_{pi} = \left(\frac{\Delta K^2}{6\pi\sigma_{yield}^2}\right) \tag{14}$$

In this model, the extent of plastically deformed metal at the crack tip is z and is updated, if no retardation occurs, as $z_{i+1} = a + r_{pi}$. During retardation, the crack grows through the plastically deformed metal, so z remains constant.

4.17 C-Implementation of Crack Growth

Cycle-by-cycle crack growth is the slowest part of the P^{2}IAT framework. The core loop takes a list of cycles (ΔK and R (stress ratio)) along with an initial crack state (length and plastic zone size) and computes the crack state at the end of the load cycles. This loop was implemented in C to speed it up. The crack growth code is a C-extension for Python which uses Python APIs (application program interfaces) directly, allowing data to be streamed in and out. Speed comparisons were done between the C-implementation and a pure Python implementation as shown in Table 2. The overhead in using the C-implementation from Python is quite small due to the use of the API, so the C version is always at least as fast as the Python version.

The Python and C-implementation of crack growth were verified to give equivalent results and to give the same results as a legacy code. Details are provided in Appendix E.

Table 2. Speed Tests for C and Python Versions of Crack Growth Engine.

Number of cycles	Exec. Time: C	Exec. Time: Python	Speedup
10	5.64E-04	6.02E-04	**1x**
100	4.99E-04	2.30E-03	**4.6x**
1,000	1.26E-03	2.02E-02	**16x**
10,000	4.49E-03	1.11E-01	**25x**
100,000	3.70E-02	9.68E-01	**26x**
1,000,000	4.54E-01	1.00E+01	**22x**

4.18 Updated Crack State

The crack growth routine outputs an updated crack state at the end of the load cycles. The updated crack state is the crack length in the surface and depth directions and the plastic zone sizes in both directions. As with all variables in P^{2}IAT, each of these parameters is a distribution described by a set of samples. Again, in the case of a through crack, the crack length in the depth direction is equal to the thickness and should remain constant. The updated crack state is used to start the next iteration of crack growth, as shown in Figure 34. In the standard data rhythm, this occurs after each set of flight recorder data is downloaded and used to update the model. Then, the crack state used to initialize the next run is no longer the EIFS distribution, but the updated crack state from the previous run. Within P^{2}IAT, and within one data download, the tool in fact performs the same operation on a flight-by-flight basis.

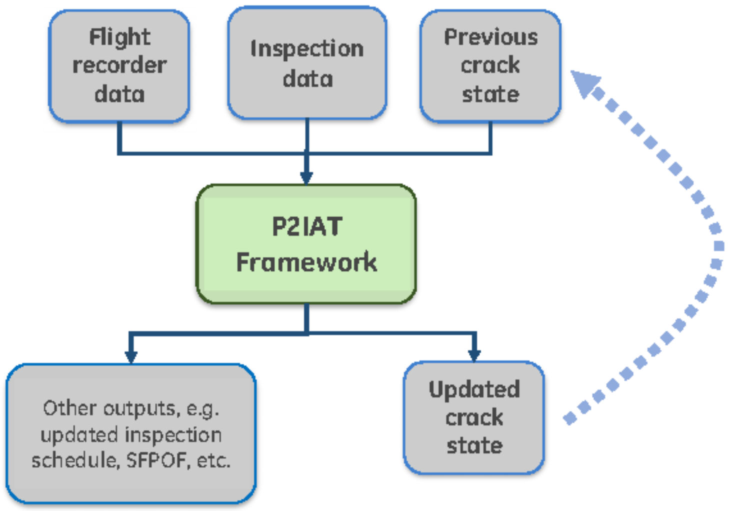


Figure 34. Data Rhythm of P^{2}IAT.

4.19 Inspection Model (POD)

The P^{2}IAT framework takes a probabilistic model of the inspection method as input. The inspection model is used for multiple purposes:

1. To perform updating with inspection results.
2. To predict inspection results in the future.
3. To schedule inspections at times when the inspection data would be most helpful in updating the model.

The P^{2}IAT framework can incorporate crack detection information to improve predictions. The ability to incorporate this information is important because many NDI (non-destructive inspection) techniques are only able to return a true/false value of crack detection and only rarely is crack length able to be measured. POD (probability of detection) models based on the handbook MIL-HDBK-1823A [24] were implemented in the framework. The POD model is specified by an S-curve, μ, σ, and an optional log transform. The probability of detection is computed as follows:

$$POD(a) = S\left(\frac{T(a)-\mu}{\sigma}\right),\ T(a) = a \text{ or } \log(a) \tag{15}$$

where $S(\cdot)$ is one of four S-curves shown in Table 3, $T(a)$ is an optional log-transform, and μ and σ are centering and scaling parameters that are determined in a POD study.

Table 3. S Functions for POD Models.

Probit	$S(f) = 1 - \Phi(f)$
Logit	$S(f) = \dfrac{e^f}{1+e^f}$
C-Loglog	$S(f) = 1 - e^{-e^f}$
Loglog	$S(f) = -e^{-e^{-f}}$

In this probabilistic framework, the model form is kept constant but the parameters μ and σ are considered to be uncertain. A POD study returns their mean values, along with standard deviations and correlations (Figure 35). This yields 5 uncertain inputs: $\hat{\mu}, \hat{\sigma}$, and the three independent values in the POD covariance matrix (i.e. the covariance between μ and σ). Figure 35 shows an example POD curve taken from the handbook which includes values for $\hat{\mu}, \hat{\sigma}$, and the covariance matrix. Note that these five inputs, along with the choice of $S(\cdot)$ and $T(\cdot)$, fully describe the POD model.

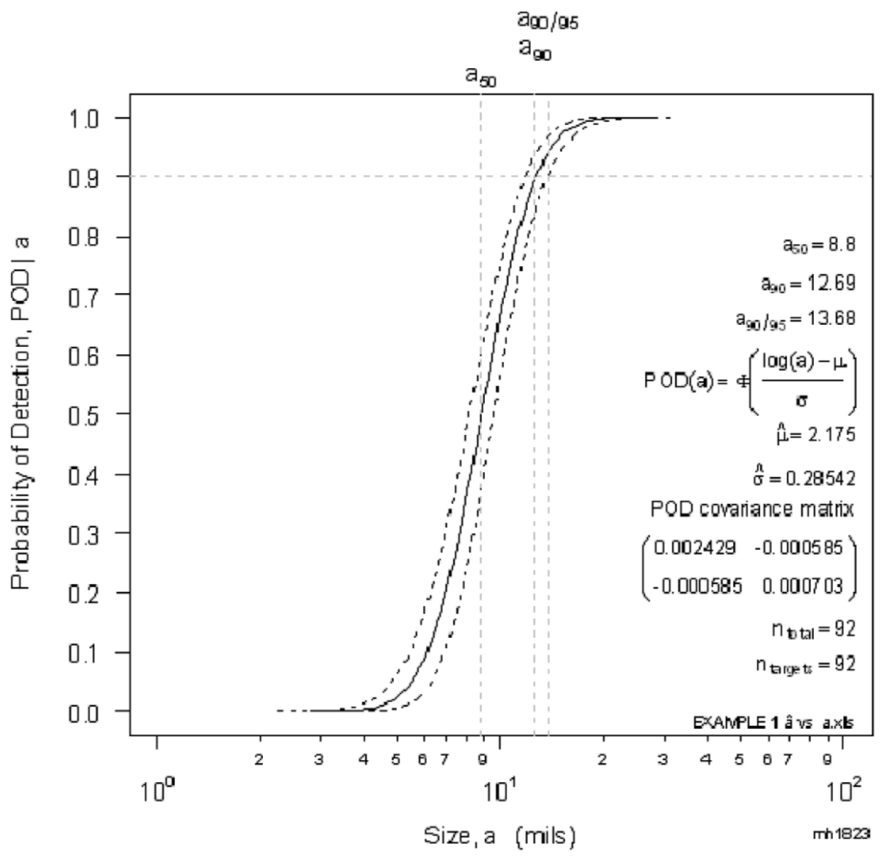


Figure 35. Example POD Curve from MIL-HDBK-1823A [25].

The inspection model in general could be formulated as a function of crack length in the depth and surface directions in the case of a corner crack. Another alternative is to build the POD model as a function of the crack face area. In an even more advanced usage of P^2IAT, the inspection model could depend on the geometry or other parameters and involve a complicated physics model (or GEBHM meta-model) of the inspection process. In this case, the Bayesian updating capability of P^2IAT would essentially perform NDI model inversion. In addition, multiple inspection techniques may be used at a single location. Multiple POD models can be implemented separately with separate inspection data for each technique.

When no inspection data is specified, the framework will predict what an inspection would return. When data are specified, the data are used to update all uncertainty in the framework (all variables specified as distributions, e.g. initial flaw size, crack length, material properties, etc.).

4.20 Critical Stress Intensity Factor

The failure criteria can be quite general in P^2IAT, but specific code has been written for using a critical stress intensity factor (also known as fracture toughness). The most basic probability of failure computation consists of comparing the stress intensity factor, K_I, with the critical stress intensity factor or fracture toughness K_{Ic}. The fracture toughness is commonly available in material testing databases, for example in [20]. From a set of sample data, a fracture toughness distribution can be fit, usually a normal distribution.

4.21 Failure Criteria (SFPOF)

Failure is assessed by comparing the fracture toughness distribution with the distribution of K_I whose uncertainty is due to load, geometry, and crack state uncertainty. Typically, probability of failure is computed as $P(K_I > K_{Ic})$ where both factors may be random variables. In the case of P^{2}IAT where probabilities on the order of 10^{-7} are desired, it is difficult to get accuracy at this level with sampling methods. By using an analytic distribution (normal) for K_{Ic}, though, small probabilities can be computed. For each sample of the load-induced stress intensity factor K_I^i, the overall probability of failure can be computed as

$$P(K_I > K_{Ic}) = \frac{1}{N}\sum_i P\left(K_I^i > K_{Ic}\right) = \frac{1}{N}\sum_i \Phi\left(\frac{K_I^i - \mu_{Ic}}{\sigma_{Ic}}\right) \tag{16}$$

where μ_{Ic}, σ_{Ic} are the mean and standard deviation of the fracture toughness distribution and Φ is the standard normal CDF. The CDF evaluation enables probabilities of failure on the order of 10^{-7} to be computed.

There are two options for what load should be used in computing the stress intensity factor for assessing the probability of failure. In one case, the expected loads (load forecast) are used. The maximum expected stress intensity for a flight goes into the POF calculation. Alternatively, one could use a constant limit load. This would yield POF numbers with the interpretation that "failure" occurs when the structure fails to fulfill its design intent, even if it may still be able to bear normal usage loads.

A complex probability of failure function taking into account the surface, depth, and through crack stress intensity factors is required for corner cracks at bolt holes in plates. Specifically, the relevant cases are as follows:

1. If depth crack length > thickness, use through crack K_I
2. If depth crack length < thickness and depth $K_I > K_{Ic}$, use through crack K_I
3. If depth crack length < thickness and surface $K_I > K_{Ic}$, use corner crack surface K_I
4. If depth crack length < thickness and surface $K_I < K_{Ic}$, use corner crack surface K_I

The final failure criteria in P^{2}IAT is the Single Flight Probability of Failure (SFPOF). SFPOF is defined as a conditional probability of failure:

$$SFPOF = P(fail\ at\ N | not\ failed\ at\ N-1) \tag{17}$$

That is, SFPOF is the probability of failure at flight N given that the asset has survived up to flight $N-1$. This is accomplished using the Bayesian updating formula and considering the survival up to flight $N-1$ as data. That is, the model is updated not only with inspection data but also with "is_failed = False" data. This of course requires defining a failure node in the network, which is described below. This technique was shown in [5] and verified to produce accurate SFPOF values in P^{2}IAT. See Appendix G for details and a verification exercise.

4.22 Fail/No-Fail

A binary node representing failure or not-failure is included in the network in P^{2}IAT so that the model can be updated with not-failure data ("is_failed = False").

4.23 Sensitivity

In general, global sensitivity methods break down a function of many variables into a hierarchy of functions, each depending on some subset of the variables [26]. Main effect functions vary only with one input, while joint effect functions vary with two (higher order effect functions are defined analogously). The global sensitivity index (Sobol index) is then:

$$S_i = \frac{V_i}{V}, \quad S_{i,j} = \frac{V_{i,j}}{V} \tag{25}$$

where V_i is the variance of the effect function for the ith input x_i and V is the overall variance of the function. In GE's BHM code, these variances can be computed analytically due to the functional form of the meta-model. However, there is no meta-model built in DBN; rather it is just a collection of samples. Thus, a new method must be developed which can ideally make use of the existing samples. Many sample-based Sobol index computation methods require one to create a large set of samples which may be computationally prohibitive.

A binning-based method was developed to compute Sobol indices based on existing samples. The method is shown schematically in Figure 36. The particles are binned with respect to the value of the input x_i (Steps 3 and 4). Within each bin, the variance of the output y is computed (Step 5). Then, the average of the per-bin variances is computed (Step 6). Finally, Step 7 computes the sensitivity index with a simple formula. This is repeated for each input and output combination desired.

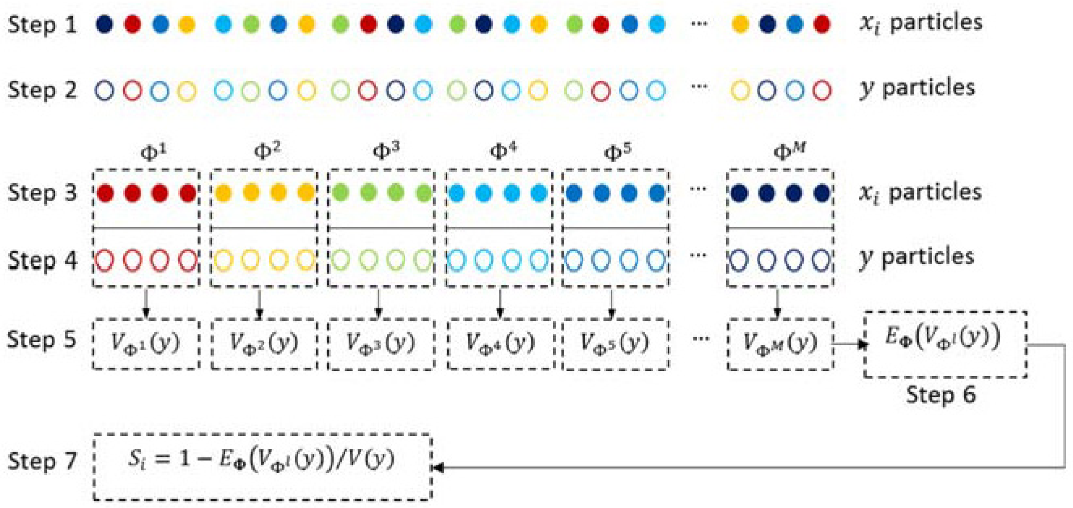


Figure 36. Schematic of Method for Computing Sobol Index for Given Samples.

There are several benefits of this method. First, the estimated Sobol indices are unbiased. Second, binning avoids any assumption about the form of the distribution of interest. Third, the estimate is relatively accurate compared to other binning methods which first compute the mean within a bin and then the variance across bins. In those methods, binning errors tend to be accentuated in the final Sobol index, whereas in the method proposed here the errors tend to cancel out.

In the P^{2}IAT framework, crack length distributions are computed after every flight. Thus, the sensitivity of crack length to an input parameter is actually a function of time. In order to summarize the overall sensitivity, while taking into account the fact that the overall variance V_i is also a function of time, the generalized sensitivity index is computed as

$$GS_i = \frac{\sum_t S_i(t)V(t)}{\sum_t V(t)} \tag{26}$$

The generalized sensitivity index can then be compared for all of the nodes in the network (crack length, material properties, load uncertainty, geometry parameters, etc.).

Sensitivity indices can be used to diagnose which input parameters are causing the most variability in one or more outputs of interest. In the case of uncertain loading, the sensitivity analysis does not apply directly because one cannot determine bins for a large vector like load history. One could consider creating a few scalar variables which represent characteristics of the load vector (e.g. mean, median, max), or leave it out from the sensitivity analysis altogether. By default in P^{2}IAT, only scalar parameters are used in the sensitivity analysis.

4.24 IAT Reports

The P^{2}IAT framework also includes routines to generate IAT reports in a format agreed to by the Air Force and the team. A control point summary table gives distribution of the crack length on the surface by specifying the 10%, 25%, 50%, 75%, and 90% percentiles of the distribution. Each of these percentiles also has an upper and lower 95% confidence bound computed using bootstrapping. The next maintenance action for each control point is included. The spectrum severity factor is also included which compares the crack length to a baseline projection.

The maintenance summary table gives maintenance packages for upcoming inspection times. At each potential inspection point, all control points which are scheduled to have an inspection are listed along with the type of inspection (e.g. field-level or depot-level). At the conclusion of Task Order 0001, the reports were only generated for a single control point and one type of inspection.

The final IAT report is the SFPOF curve over time. A plot of the overall SFPOF is generated which shows how the risk is reduced at any points where inspections have occurred. The forecasted SFPOF may also reduce at scheduled inspection points.

Some details regarding the software implementation of P^{2}IAT are described in Appendix D.

5.0 INSPECTION CRITERIA

The P^{2}IAT framework can use several different methods for scheduling inspections. The simplest is to perform inspections at fixed intervals (e.g. every 200 flights). A more advanced method is to perform inspections at a specified risk threshold. For example, the inspection can be set whenever SFPOF exceeds a threshold. This is quite simple to implement as SFPOF is already calculated at every flight. A third method is to use an information gain criteria. The information gain criteria attempts to place inspections at times when the inspection result is the most informative. A detailed discussion of many possible inspection scheduling methods and how they might be implemented, along with some examples, is given in Appendix F. The final method of using information gain implemented in P^{2}IAT is discussed in the following paragraphs.

The P^{2}IAT framework, and specifically the DBN within it, can use various forms of field data for updating. In addition, one requirement of the framework is to schedule future inspections. It is worth considering how inspection information is used as this can lead to optimizing the inspection intervals. Note that we will not be formally setting up or solving an optimization problem.

In the DBN, inspection information is incorporated by performing Bayesian inference on all nodes in the network. That is, the network's predictions at a given point in time are considered the prior distribution. Then, a Bayesian inference step (i.e. weighting by the likelihood) is performed based on the data at that time. The result is a posterior distribution for all the network parameters. If the data are very noisy, then the posterior distribution will be nearly identical to the prior. In this case, the data did not inform the model very much. Alternatively, if the data are of high precision (e.g. crack length measurement with accuracy of <0.001 inch), then the posterior distributions are either shifted or narrowed (or both). In this case, the data had a significant impact on the model.

Data that has a bigger impact on the model are more valuable, and therefore we would prefer to perform inspections at those times. Thus, a measure of this impact is needed. We have chosen to use the Modified Kullback-Leibler (KL) divergence [27] to measure the difference in information between the prior and posterior distributions. The KL divergence is computed as

$$D_{KL} = \int p(a) log \frac{2p(a)}{p(a)+q(a)} da + \int q(a) log \frac{2q(a)}{p(a)+q(a)} da \qquad (27)$$

where $p(a)$ and $q(a)$ are the probability distributions of the crack length from the model predictions and the measurement, respectively. In the case of crack detections, the POD curve essentially serves as $q(a)$. The modified form simply re-arranges the equation to aid in computations using sampled distributions. The KL divergence reduces to zero when the two distributions are identical and approaches infinity when there is no overlap between the distributions. This situation is very unlikely when applied to the DBN which is solved with a particle filter. Thus, a higher value of the KL divergence implies that more information was learned from the data. In fact, the units of the KL divergence are bits of information.

Applying the KL divergence to the P²IAT framework requires a choice of variables for which it is computed. That is, there may be many variables that are updated when some particular piece of data is incorporated into the model. It is certainly possible to examine the information gain in many variables. However, the crack length distribution is arguably the most important parameter in the model because it strongly governs all of the failure criteria. Thus, in the P²IAT framework information gain will refer to a large KL divergence (D_{KL}) of the crack length distribution when data are incorporated into the network.

To schedule inspections, we begin by taking a greedy approach. That is, we look only into the near future to find the next optimal inspection. To do this, the crack growth is forecasted into the future. At each possible time point in the future, a measure of the expected information gain is computed. This is done by assuming a particular inspection datum and computing the KL divergence between the prior (forecasted) crack length and posterior (after updating with the assumed datum) crack length distribution. This results in a value of the KL divergence at each time point. Finally, the time for the next inspection (in cycles or flights) is determined as the time at which D_{KL} is maximum.

In terms of IAT, only crack detection information will be considered. It is likely that crack detections will be the only available data from an inspection. The POD curve has a very strong effect on D_{KL} and thus the computed inspection time. Suppose the point POD=0.5 occurs at the crack length a_m (the middle of the POD curve), see Figure 37. If the (forecasted) crack length distribution is very small compared to a_m, then the inspection will almost definitely return "not detected." The "not detected" information will not significantly change the crack length distribution, as it was already known to be undetectable. On the other hand, if the (forecasted) crack length distribution is very large, then the inspection will almost definitely return "detected." Again, very little information is gained because the crack length distribution will remain high. Only if the crack length distribution is near a_m will the detection data have a significant impact on the DBN.

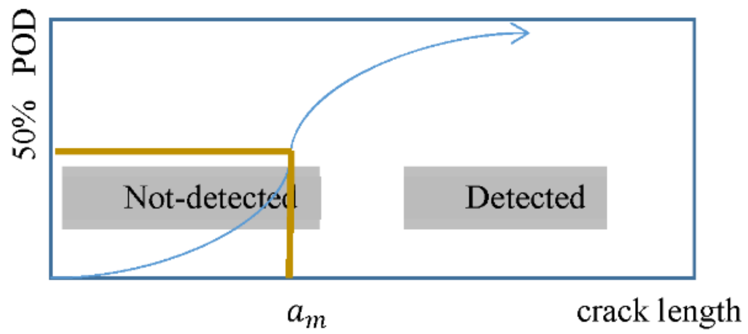


Figure 37. Inferred Crack Sizes from POD Curve.

In addition, different types of inspections, e.g. eddy current vs. ultrasonic, have different POD curves, so they can be easily compared in terms of information gain. In general, the information gain of any number or type of inspection can be compared. The overall process is:

1. Forecast crack length.

2. At some time, use the forecasted crack length and a given inspection method (POD curve) to create simulated inspection data. One simulated inspection result is created for every particle.
3. Temporarily update using the data. A single draw from the updated model is associated with the particle that was used to generate the inspection result. Thus, all possible inspection results are taken into account in the expected update.
4. If considering repairs, the updated model may be re-set to the as-repaired state (e.g. re-setting the crack length to the EIFS distribution and potentially updating some model parameters like bolt hole radius to consider an oversize fastener).
5. Compute D_{KL} between prior (forecast) and posterior (updated) crack length.
6. Discard updates and continue forecasting.

The process results in a curve $D_{KL}(t)$ for each inspection method. This is the information gained by performing the inspection at any particular time. Computing the information gain is especially helpful for inspections where the inspection method is only accurate for a small part of the life of a part. Then, the inspection will be scheduled for the time when the inspection is most accurate. One can use the information gain to compare inspection methods and even incorporate a cost estimator to consider, e.g., information gained per dollar spent. This could further help make maintenance decisions with various qualities of inspection.

The user may specify a number of options in P^{2}IAT to make particular assumptions during inspection scheduling. These include:

- Use or do not use simulated inspection results for updating
- Repair/do not repair when the crack is detected or not detected
- Restrict inspections to a particular interval

5.1 Uncertainty and Risk

An important consideration in the definition of inspection schedules based on risk is the inaccuracy in the models. Model inaccuracy is reflected in the distribution of crack length. Thus, if the models are very inaccurate, the crack length distribution will be very wide. A wide crack length distribution often leads to a significant probability of failure, since large cracks are possible. This probability of failure, though, is due more to the lack of knowledge in P^{2}IAT rather than an accurate prediction of failure. If the P^{2}IAT framework does not have much data, the predicted uncertainties will be large.

If one were to design inspection schedules based on a model that has little data, the result would be constant inspections (and model updating) until the model predicts narrower crack length distributions. This strategy, however, may be unacceptable to the users who may need to move forward with little data. One way to get more reasonable inspection schedules with an ill-informed model is to trust the mean of the model. This could be done, for example, by artificially reducing the variance of the initial flaw size (or of the crack length distribution), the variance in the applied loads (e.g. forecast usage by just repeating a single flight), or the variance in other sources of uncertainty (e.g. material properties, load conversion, etc.). Note that it is quite simple to modify these distributions in the P^{2}IAT framework.

6.0 SINGLE POINT DEMONSTRATION

The SAFER- P^{2}IAT framework is demonstrated on a single control point in the wing of a fighter aircraft. The demonstration consists of developing the baseline usage data and all models which are used in the P^{2}IAT process (see Figure 17). A crack at the demonstration location is grown probabilistically using baseline usage data. A simulated inspection is performed and the data is used for updating, resulting in reduced uncertainty in the parameters. Crack growth is then forecasted using the most up-to-date estimates of the parameters, crack state, and usage model. Inspection scheduling is performed using the maximum information gain criteria, and finally IAT reports are generated.

6.1 Control Point Definition

The control point used in the demonstration is a bolt hole in the outboard section of the right wing of a fighter aircraft. The bolt connects the lower forward wing skin to a rib, near the front spar. The bolt is shown in Figure 38. The location was chosen based on cracks found in full scale fatigue tests. Cracks were found in the top corner of the wing skin (i.e. at the faying surface between the wing skin and the rib lower cap). The crack is assumed to grow in the wing skin at a location determined by the local, fine-grid Finite Element model.

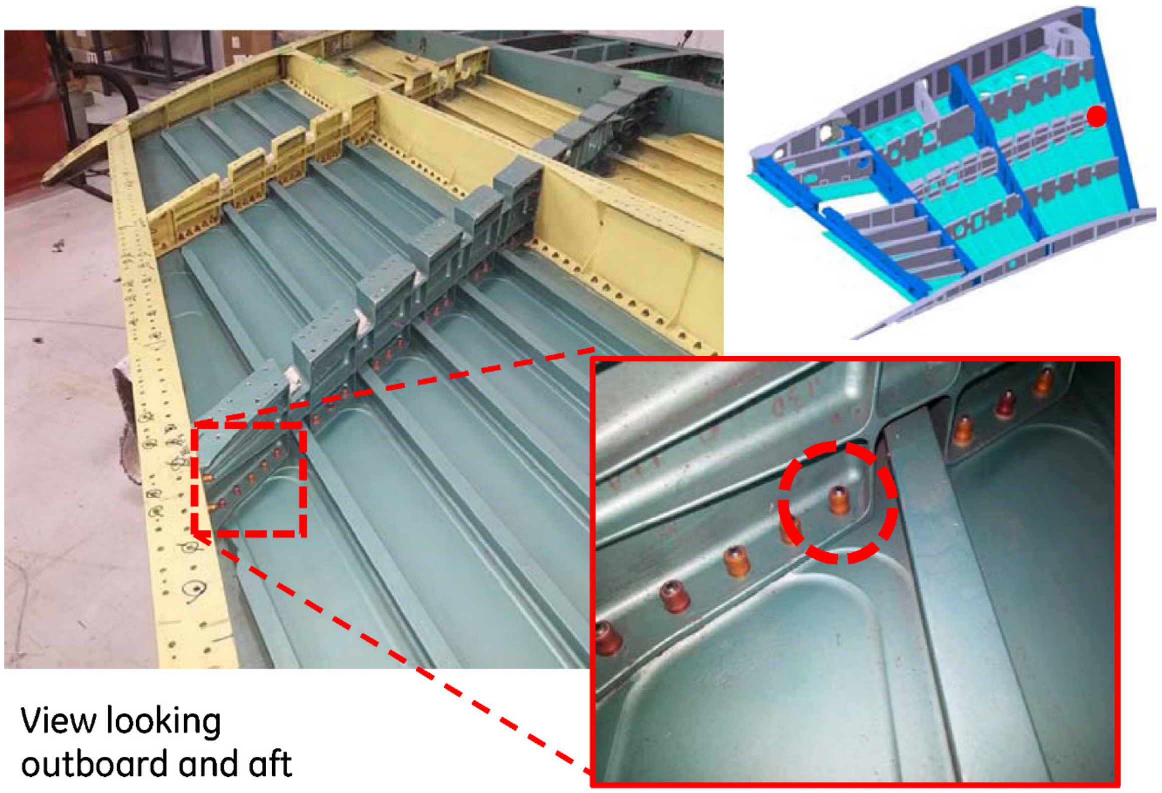


Figure 38. Control Point Used for P^{2}IAT Framework Demonstration.

The entire P^{2}IAT framework was demonstrated on single location for the baseline spectrum loading. The framework flow chart in terms of data and models is shown in Figure 17. Another view of the framework, in terms of inputs and outputs, is notionally shown in Figure 39. Here, inputs are grouped by category. The single point demonstration was done for a particular set of inputs corresponding to, in general, the most physically realistic estimates for the demonstration location. The inputs and settings are discussed below.

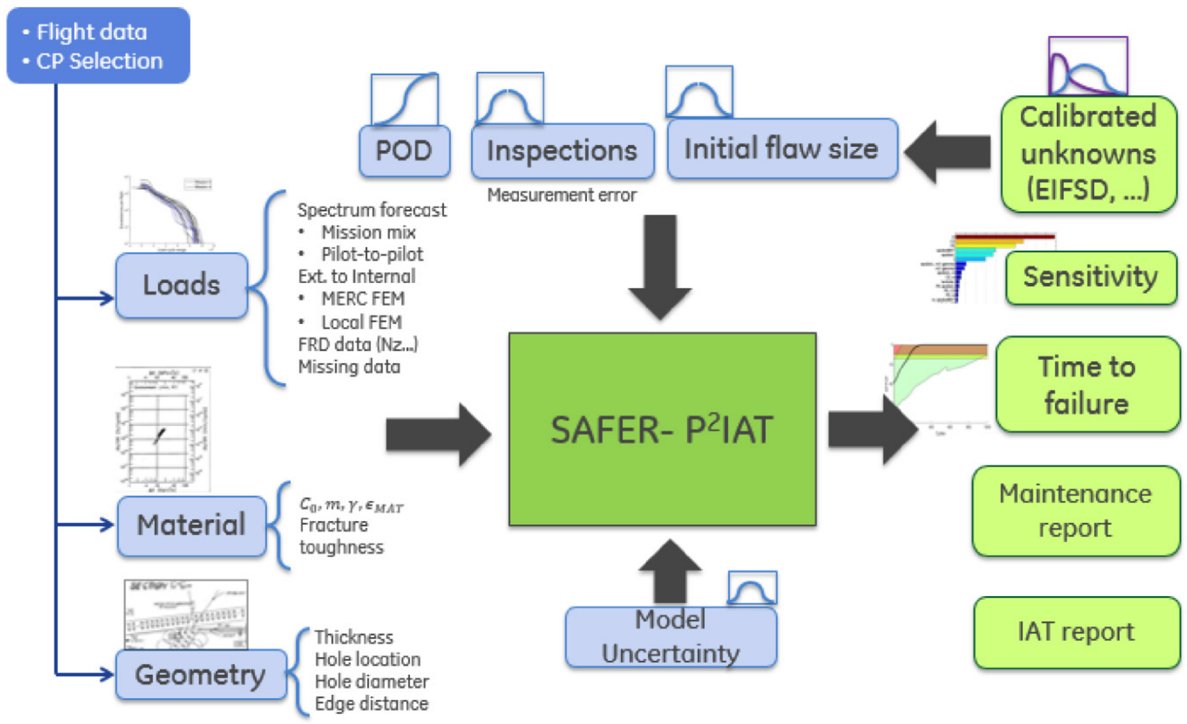
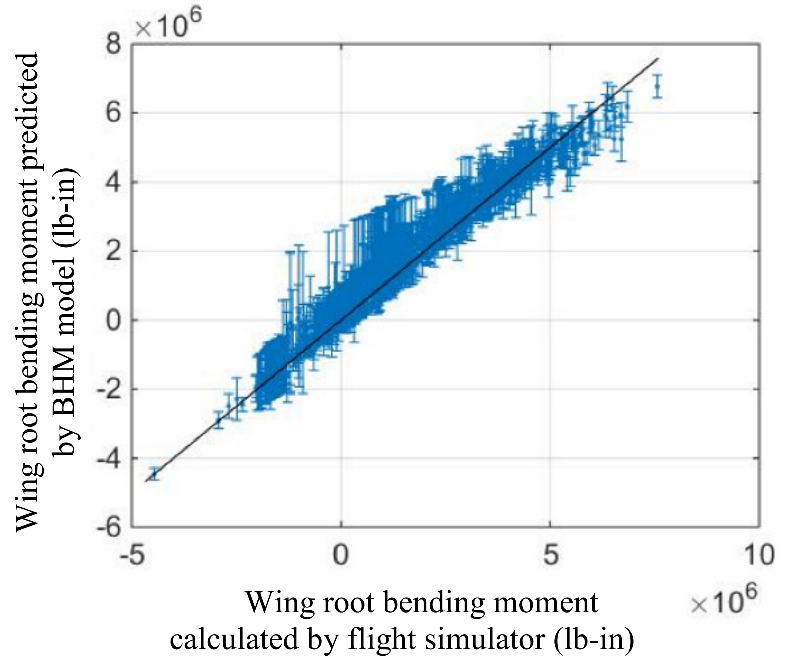


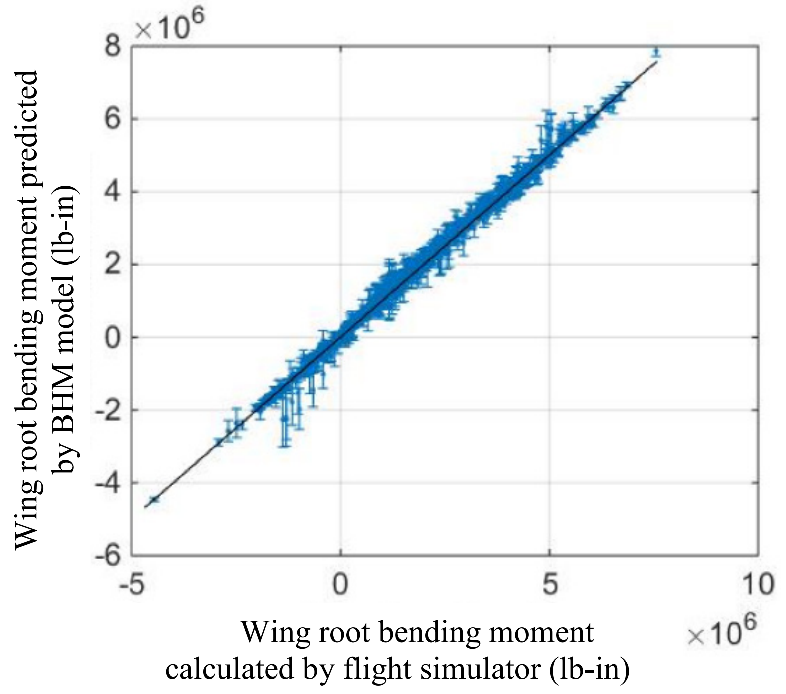
Figure 39. P^{2}IAT Framework.

6.2 Loads

The first task for the demonstration is to build the model for predicting external loads given flight recorder data using the flight simulator run database. The plots in Figure 40 show validation plots (predicted versus actual bending moments) for the flight recorder data to external loads model at iterations 1 and 10 in the IDACE process. Points shown are a representative 2,000 of the roughly 34,000 validation points. The model at iteration 1 was trained with 100 data points, whereas by iteration 10 it had 1,000 data points. The accuracy increases significantly as the IDACE iterations progress, and the final model at iteration 10 predicts external loads quite well.



a) Iteration 1 - 100 Training Points



b) Iteration 10 - 1,000 Training Points

Figure 40. Validation Errors on 2,000 Points not used to Build Model.

Next, the loads actually applied in P^{2}IAT must be generated. The loads consist of drawing from the baseline MES (noting that the loads of interest are the bending and torque on the right-wing). The single point demonstration consists of two runs. First, a baseline forecast using the

MES and baseline mission mix is done. Second, a data assimilation task is performed where 100 flights of "data" are used to grow the crack, followed by updating using inspection data and then a 1,000 flight forecast with inspection scheduling. The 100 flights of data are taken as the actual flight data. Note that only flight recorder data and mission ID are used as inputs to P^{2}IAT (not external loads or jack loads). In addition, flight recorder data from 5 flights are removed (missing data) while the mission ID is kept. The flight recorder data is converted to external loads (a value for bending and torque) as a pre-processing step.

After the data is assimilated and the model is updated, forecasting proceeds by drawing flights from the MES under the current mission mix. Each sample or particle draws a different mission and flight. Note that built into the MES are uncertainty due to mission and pilot-to-pilot variation, which is accounted for by having a number of flights which are different ways of flying the same mission.

In summary, the uncertain inputs relating to loads are:

- Mission type/mix
- Pilot-to-pilot variation
- Missing data
- Flight recorder data to loads conversion $\epsilon_{FRD \to load}$

6.3 Stress Intensity Factor Model

Probabilistic values of K_I were computed for a number of parameter sweeps to check if the trends were physically meaningful. Uncertainty due to the Green's function and stress models were included. Geometrical parameters that were varied are as follows:

- The crack lengths in the surface direction (c) and the depth direction (a)
- The hole radius (denoted R, not to be confused with the stress ratio)
- The hole offset in the x and y global aircraft coordinate system (dx and dy)
- The edge distance (r_b)

Of course, stress intensity factors were also computed for various loading conditions parameterized by the wing bending moment and torque at station 3 (WBR3 and WTR3, respectively). The trends in Figure 41 to Figure 44 for corner crack K_I are to be expected from physical intuition.

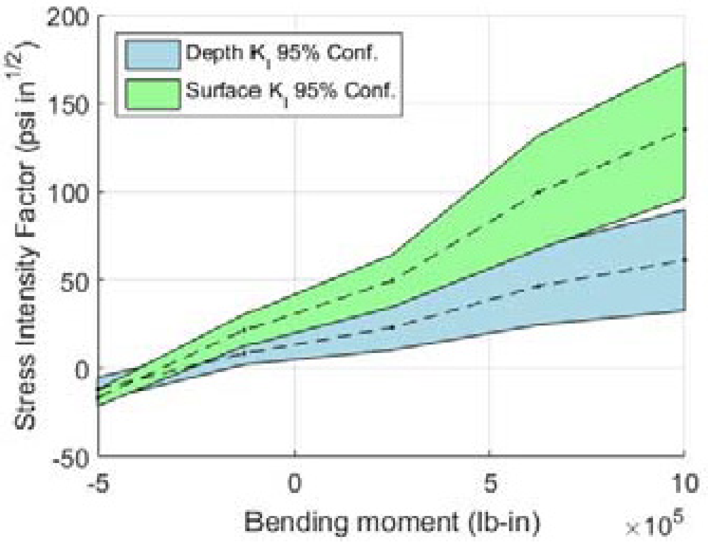


Figure 41. Probabilistic Stress Intensity Factor and Confidence Bounds versus Bending Moment. Crack size is set to $a = c = 0.01$ inches. Hole diameter is 0.25 inches. Hole offsets are $dx = dy = 0$. Wing torque at station 3 is -300,000 lb-in.

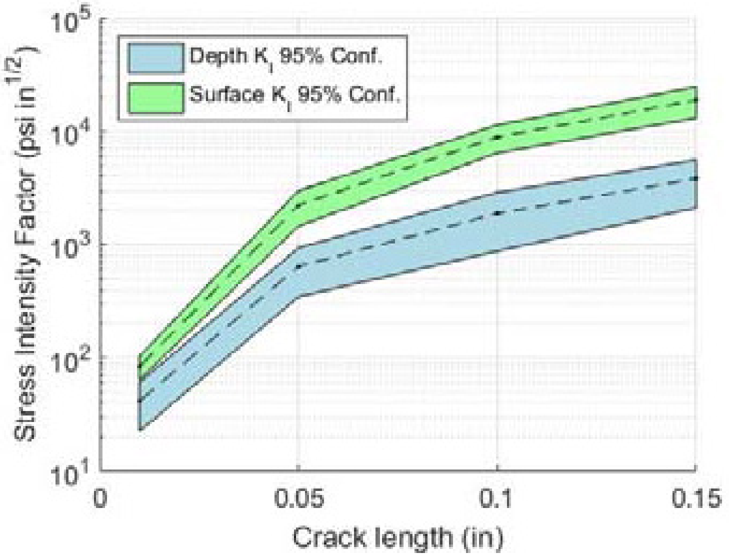


Figure 42. Probabilistic Stress Intensity Factor and Confidence Bounds versus Crack Length. Crack shape is quarter-circular ($a = c$). Hole diameter is 0.25 inches. Hole offsets are $dx = dy = 0$. Wing bending moment at station 3 is 500,000 lb-in and wing torque at station 3 is -300,000 lb-in.

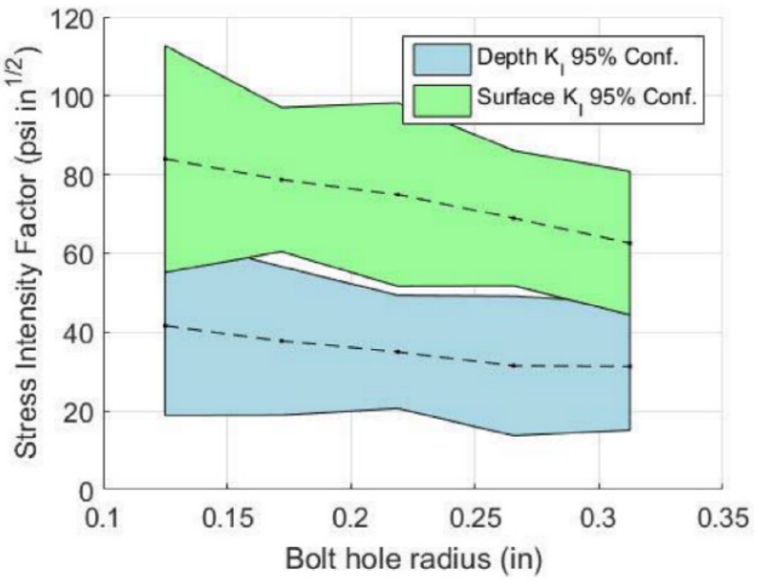


Figure 43. Probabilistic Stress Intensity Factor and Confidence Bounds versus Bolt Hole Radius. Crack size is set to $a = c = 0.01$ inches. Hole offsets are $dx = dy = 0$. Wing bending moment at station 3 is 500,000 lb-in and wing torque at station 3 is -300,000 lb-in.

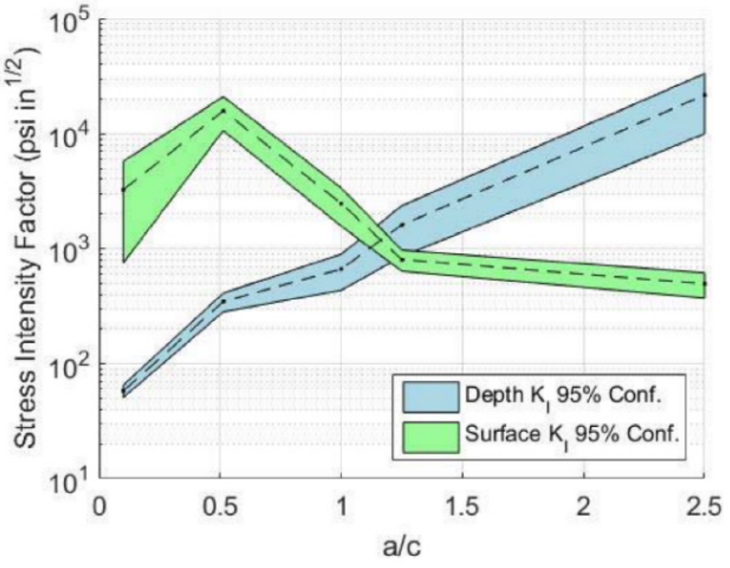


Figure 44. Probabilistic Stress Intensity Factor and Confidence Bounds versus Crack Aspect Ratio. The surface crack length is fixed at $c = 0.055$ inches and the depth a is left to vary to produce the aspect ratios shown. Hole offsets are $dx = dy = 0$. Wing bending moment at station 3 is 500,000 lb-in and wing torque at station 3 is -300,000 lb-in.

A BHM was also built for the through-crack stress intensity factor. This model had much lower uncertainty in the predictions than the corner crack model because the Green's function was analytical rather than coming from an ANN model. The comparison of the stress intensities calculated with the BHM to those from the Green's function in Figure 45 shows good accuracy. The sensitivity plot in Figure 46 shows expected trends. The crack depth parameter a is not an input since it is fixed at the material thickness for the through crack case.

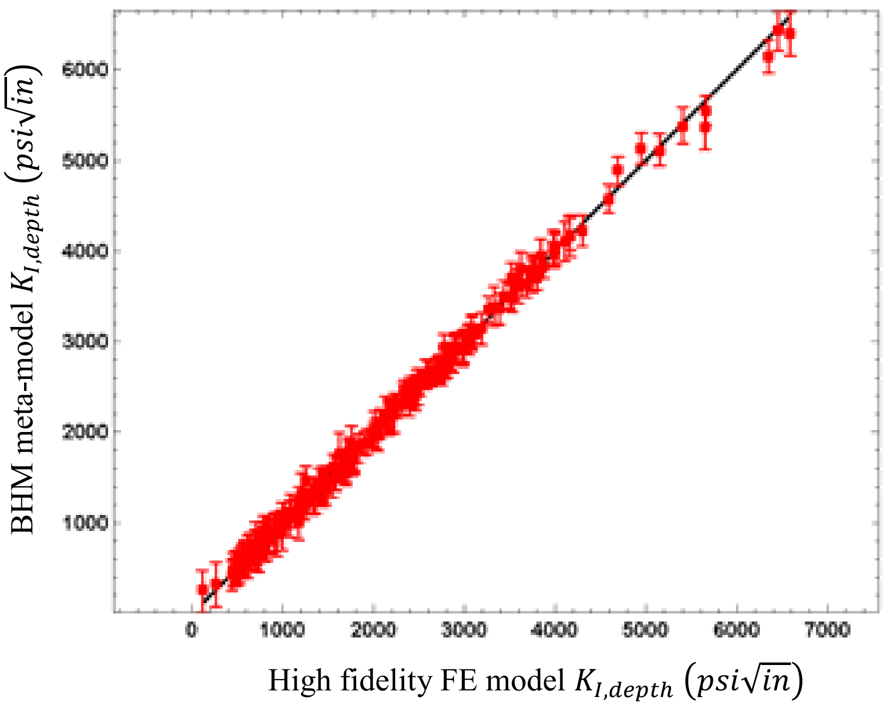


Figure 45. BHM Calculations of Stress Intensity Factor for Through Crack versus Green's Function Calculations.

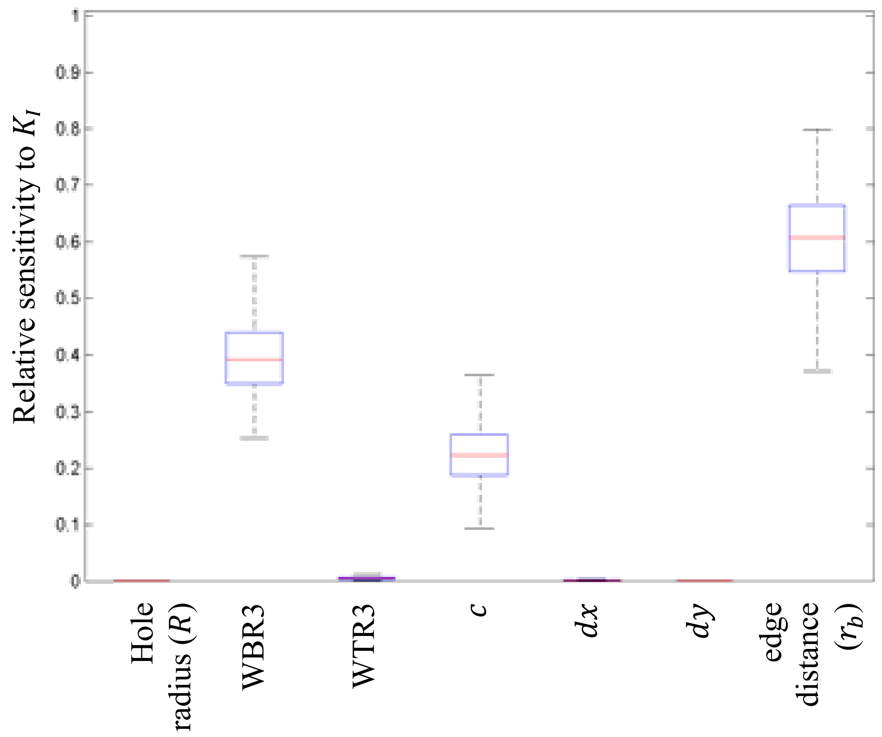


Figure 46. Sensitivities of BHM Calculated Through Crack Stress Intensity Factor

It was found that the K_I models are in fact quite simple response surfaces. While BHM models have the capability to model very complex, nonlinear behavior, they can also reduce to simple linear or quadratic models when the data warrants it. In this case, the K_I behavior is quite well represented by quadratic models of load and crack size. Geometry variables had an impact less than the model error term. General BHM models can be expensive to evaluate, so new quadratic K_I models were built. The quadratic models are about eight times faster to evaluate than the BHM models and have errors on the same order as the BHM prediction errors. The quadratic models were packaged with an additional source of uncertainty corresponding to the model error. This replaces the BHM prediction uncertainty. The model error is an additive normal random variable, $\epsilon_{load \to K_I}$, which is sampled as another unknown or uncertain input in the DBN. That is, we treat the model error as another uncertain input in the same way that, say, an uncertain material property would be treated. The sampled value of $\epsilon_{load \to K_I}$ is either positive or negative and is simply added to the quadratic K_I model prediction (in fact added to $\log(K_I)$). Note, the specific form of the quadratic is:

$$\log(K_I) = f_{quadratic}(WBR3, WTR3, \log_{10}(a), \log_{10}(c)) \tag{28}$$

where WBR3 and WTR3 are the outboard bending moment and torque and a and c are the depth and surface crack lengths, respectively.

In particular, the quadratic form for the corner crack model is

$$\begin{aligned}\log(K_{I,depth}) = {} & 11.04 - 4.66w^2 + 2.25 \times 10^{-7}wx + 2.08 \times 10^{-7}wy + + 8.77wz + \\ & 8.62w + 1.08 \times 10^{-12}x^2 + 5.17 \times 10^{-13}xy - 2.03 \times 10^{-7}xz + 3.03 \times 10^{-7}x - \\ & 3.96 \times 10^{-13}y^2 + 8.42 \times 10^{-8}yz + 3.05 \times 10^{-7}y - 4.39z^2 - 5.44z\end{aligned} \tag{29}$$

$$\begin{aligned}log(K_{I,surf}) = {} & 12.482 - 4.02w^2 + 4.19 \times 10^{-8}wx - 6.48 \times 10^{-7}wy + 7.73wz - \\ & 6.04w + 1.03 \times 10^{-12}x^2 + 5.5 \times 10^{-13}xy - 2.5 \times 10^{-7}xz - 1.72 \times 10^{-7}x - \\ & 8.06 \times 10^{-13}y^2 + 6.78 \times 10^{-7}yz - 9.8 \times 10^{-8}y - 3.95z^2 + 9.65z\end{aligned} \tag{30}$$

$$\epsilon_{load \to K_I} \sim Normal(0, 0.411) \tag{31}$$

where the variables are defined for simplicity as:

$$x = WBR3, y = WTR3, z = \log_{10}(a), w = \log_{10}(c) \tag{32}$$

The model for through-thickness cracks is:

$$\begin{aligned}\log(K_{I,through}) \\ = {} & -0.79x_n + 0.48y_n + 5.44z_n + 0.5w_n + 2.86x_n^2 - 1.55x_ny_n + 1.19x_nz_n \\ & + 0.03x_nw_n + 0.8y_n^2 - 1.36y_nz_n - 0.64y_nw_n + 0.33z_n^2 - 0.23z_nw_n + 0.01w_n^2\end{aligned}$$

Where the normalized variables are defined as:

$$x_n = \frac{x + 2.95e5}{1.495e6}, y_n = \frac{y + 3.5e5}{6.97e5}, z_n = \frac{z + 3.29}{5.29}, w_n = \frac{w - 2}{2.98}$$

(33)

$$\epsilon_{load \to K_I} \sim Normal(0,0.15) \tag{34}$$

where in this case the variables are:

$$x = WBR3, y = WTR3, z = \log_{10}(c), w = edge\ distance \tag{35}$$

In all of these equations, the bending moment and torque are measured in inch-pounds and distances are measured in inches.

6.4 Material Properties

The material at the demonstration location (i.e. the wing skin) is Al 2124-T851 plate. A Walker model was fit to crack growth rate data from the Damage Tolerant Design Handbook [20]. The Walker model fits a crack growth curve that includes the stress ratio effects:

$$\frac{da}{dN} = \frac{C_0}{(1-R)^{m(1-\gamma)}}\, \Delta K^m \cdot \epsilon_{MAT} \tag{36}$$

The coefficients of the model were fit using linear regression in the log space. The result is a correlated multivariate normal distribution for the three coefficients, along with an independent normal variable representing the residual or model error term. Here the error term ϵ_{MAT} is additive in the log-space, where the linear model is fit. The result is a multiplicative model error in the da/dN space. Values for the model parameters are shown below.

$$logC_0 \sim Normal(-8.63,0.0817) \qquad m \sim Normal(3.211,0.078)$$

$$m(1-\gamma) \sim Normal(0.720,0.129) \tag{37}$$

$$Corr(logC_0, m) = 0.406 \qquad Corr\big(logC_0, m(1-\gamma)\big) = 0.973$$

$$Corr\big(m, m(1-\gamma)\big) = 0.247 \tag{38}$$

$$\epsilon_{MAT} \sim Normal(0,0.035) \tag{39}$$

Load interaction effects were modelled with a Wheeler retardation model. This model has two additional inputs, the yield stress (set to $\sigma_{yield} = 68.0$ ksi) and retardation exponent ($r_e = 1.3$), which were kept fixed for this demonstration.

Fracture toughness is an important material property to accurately characterize since it plays a key role in computing the probability of failure. 16 data points from [15] were considered which corresponded to the manufacturers Alcoa and Reynolds. Only data for the L-T orientation is used, since no data for L-S was available. The crack growth in the surface direction corresponds to the L-T orientation, whereas the depth direction growth would correspond to the L-S orientation. Typically, in the absence of L-S data, L-T data is assumed to be sufficient for both.

The resulting fracture toughness distribution is normal with a mean of 29 ksi$\sqrt{\text{in}}$ and a standard deviation of about 1.9.

In summary, the uncertain inputs relating to material properties are:

- Fatigue crack growth properties C_0, m, γ
- Fatigue crack growth model error ϵ_{MAT}
- Fracture toughness $K_{Ic} \sim Normal(\mu_{Ic}, \sigma_{Ic})$

6.5 Geometry

Most of the geometrical information is encapsulated in the BHM model which computes stress intensity factors from external load. This includes the air vehicle FEM, local fine grid FEM, and Green's functions used to build the K_I models. The model uncertainty in the K_I model captures the model errors in all of those analyses. While building the K_I models, though, some parameters were exposed so that their impact can be assessed. The parameters left exposed are shown schematically in Figure 47. The edge distance, which affects the Green's function, is set as a uniform distribution between 0.49 and 0.51 inches. The radius of the hole, which affects both the Green's function and the local FE model (modelled by morphing the local FE mesh), is set as a uniform distribution between 0.0938 and 0.094 inches. The lower bound is the nominal radius of the hole and the upper bound is the radius of one oversize fastener. Note the fastener at this location is a Hi-lok HL11. The offset of the hole center, which affects the local FE model, is set as a uniform distribution between -0.01 and 0.01 inches for both the horizontal and vertical offsets.

The stress intensity factor models used for the demonstration were those built for stress at the bottom surface of the skin, the location of maximum stress in the local FE model for most load cases. The quadratic models in section 6.3 were used in the demonstration.

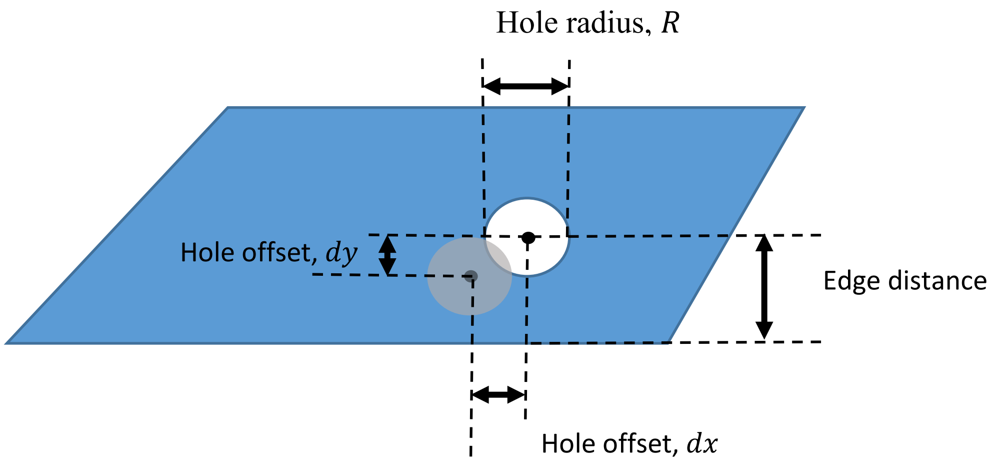


Figure 47. Geometry Parameters for the Demonstration Location.

The plate thickness, 0.13 inches, was kept constant for the demonstration. The thickness is used to switch between corner and through crack K_I models depending on the crack length in the depth direction. Each sample is able to switch between corner and through crack at any point in time; in other words, at a given time step, some samples may be corner cracks and some may be through cracks.

In summary, the uncertain inputs relating to geometry are:

- Model uncertainty from FE and Green's function $\epsilon_{load \to K_I}$
- Edge distance r_b
- Hole radius R
- Hole offset dx, dy

6.6 Initial Flaw Size

The initial flaw size was initially set using the calibrated distribution combining S-N and da/dN data. This distribution is generally quite small and the loads at demonstration location are relatively benign. This resulted in an initial forecast that no crack would form even after 40,000 flights; far beyond what would be simulated in the test. Thus, the best guess for what will happen is that no cracking will occur.

However, for the purposes of the single point demo, a larger initial flaw size distribution is created to demonstrate the capabilities of P^{2}IAT. The larger initial flaw size used for the demonstration is $a_0, c_0 \sim Lognormal(-1.6, 0.05)$. The distribution was chosen so as to demonstrate the capabilities of the tool to schedule inspections. In addition, the definition of repair was to set the flaw size distribution back to the as-built distribution. This was set as the calibrated initial flaw size from laboratory data, approximately $a_{0,repair}, c_{0,repair} \sim Lognormal(-8.4, 0.4)$. Note that when an inspection is scheduled, repair may or may not occur depending on the chance of finding a crack. The estimated repaired crack size distribution after an inspection is a combination of the as-built distribution and the current distribution, combined using the estimated probability of detecting a crack. Initially the plastic zone sizes are set equal to the crack lengths (surface and depth), and at repair they are also reset to the as-repaired crack lengths.

In summary, the uncertain inputs relating to initial flaw size are:

- Initial flaw size distribution (samples) comprising as-built and previous usage a_0, c_0
- Repair flaw size distribution $a_{0,repair}, c_{0,repair}$

6.7 Inspections

A nominal POD curve for inspections was generated. From initial coupon tests, it is apparent that a 0.150 inch crack is easily detected while a 0.050 inch crack is marginally detectable. The nominal POD curve has $a_{50} = 0.100$ inches, with a width of about 0.010 inches. For the demonstration, it is also chosen that the surface crack length (c) will be used to determine the detection probability, $POD = POD(c)$. The POD curve coefficients are uncertain, as would be obtained from a calibration exercise. The uncertainty (including the correlation matrix) as well as the specific form of the POD curve are set based on an example in MIL-HDBK-1823 [25]. The curve is defined by:

$$POD(c) = \Phi\left(\frac{\log(c \cdot 10^{-3}) - m_{POD}}{s_{POD}}\right), \quad m_{POD}, s_{POD} \sim BivNormal(\mu_{POD}, \Sigma_{POD}) \tag{40}$$

Where $\mu_{POD} = 4.6, 0.3$ (units are log(inches*1e-3)), the standard deviations are 0.05, 0.0265 and the correlation coefficient is -0.14.

The demonstration is run for two scenarios, one in which the inspection data after 100 flights is that a crack was detected, and one in which it is not detected. When the crack is detected, the updated model will tend to have crack sizes larger than 0.100 inches. When the crack is not detected, the updated model will have smaller crack sizes. An example of this behavior is shown in Figure 48 and Figure 49.

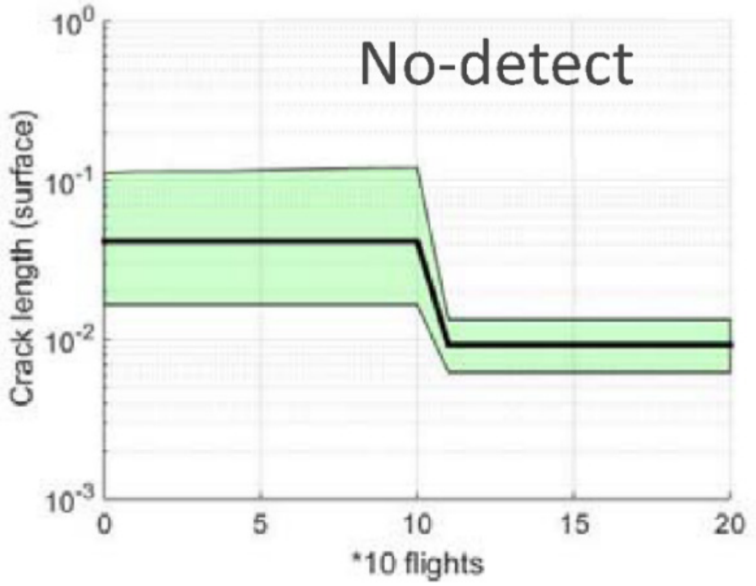


Figure 48. Example of Crack Size when Updating After No-Detect Inspection Result.

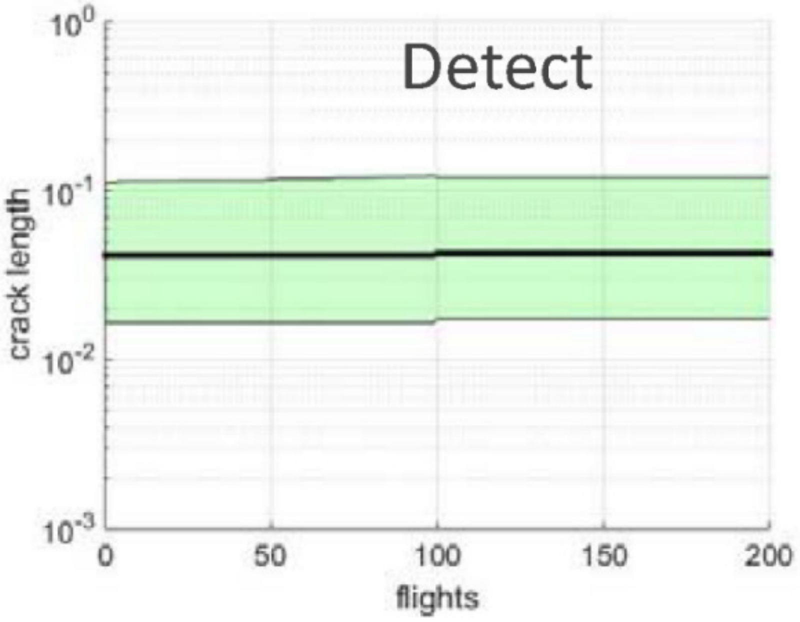


Figure 49. Example of Crack Size when Updating After Detect Inspection Result.

In summary, the uncertain inputs relating to inspections are POD curve parameters m_{POD}, s_{POD}.

This concludes all of the uncertain inputs which are fed in to P^{2}IAT for the single point demo. The framework propagates all of these sources of uncertainty through crack growth to compute probability of failure and perform inspection scheduling.

6.8 Baseline Forecast

A number of fixed settings are used for the demonstration, listed below.

- 45 processors, 1,000 samples
- Risk threshold = 10^{-7}
- Risk accumulator = mean
- Required interval = 200 flights
- No fixed interval, use risk-based scheduling and use the projected update from the projected inspection data
- Repair on detect, do not repair on no-detect
- Repair $a = a_{0,repair}, c = a_{0,repair}$
- No. of flight per batch for data = 1, forecast = 20, forecast for 100 batches (2000 flights)
- Limit load: bending moment = 1.2×10^6 in-lb, torque = -3.5×10^5 in-lb
- No truncation, upper bound on crack length = 100 inches
- Reporting period flights = 100 flights

The crack length distribution over time is shown in Figure 51. The upper and lower bounds are 10% and 90% intervals. Inspections were scheduled at 200, 400, 600, 1000, and 1800 flights, causing a decrease in the estimated crack length. The decrease is due to the possibility of finding (and repairing) a crack. Note that the distribution does not fully revert to the repaired crack size (about 0.0002 inches) until the third inspection. This is because there is a low chance of detecting the crack at the first inspections, even though the risk of failure is high. A large risk of failure

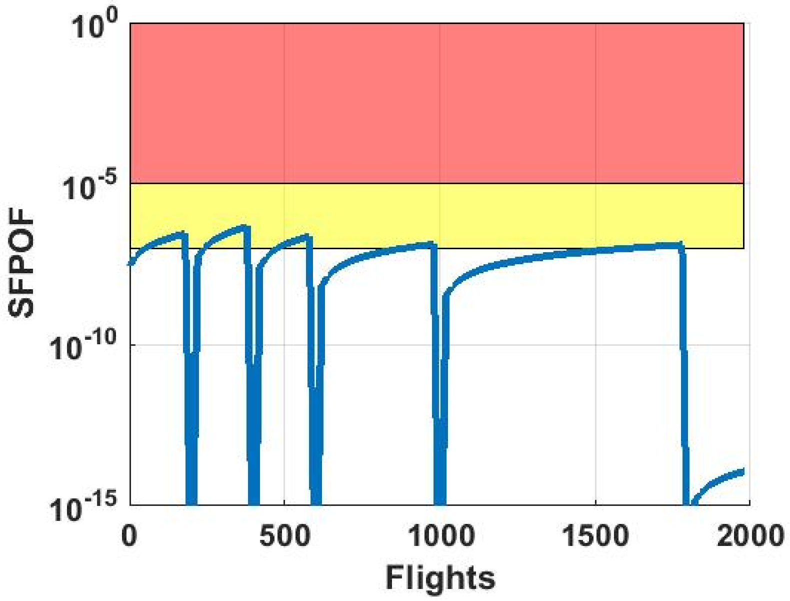


Figure 50: Probability of failure for the baseline forecast.

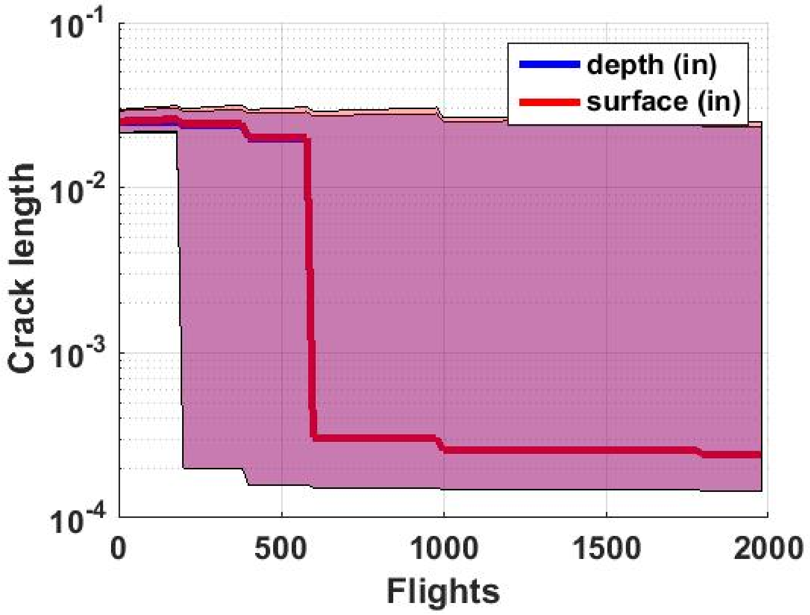


Figure 51: Crack length distribution for the baseline forecast.

($\text{SFPOF} > 10^{-7}$) can occur while the probability of detection is low because the SFPOF is highly sensitive to the far upper tails of the crack length distribution. At the same time, probability of crack detection is largely a function of the median crack.

The probability of failure is shown in Figure 50. The yellow band is between a failure probability of 10^{-7} and 10^{-5}. Inspections are scheduled whenever the SFPOF will reach 10^{-7} within the next 200 flights, even if it is low at the time of inspection. The large downward spikes are due to moderate size cracks that are inferred by updating and quickly grow to failure.

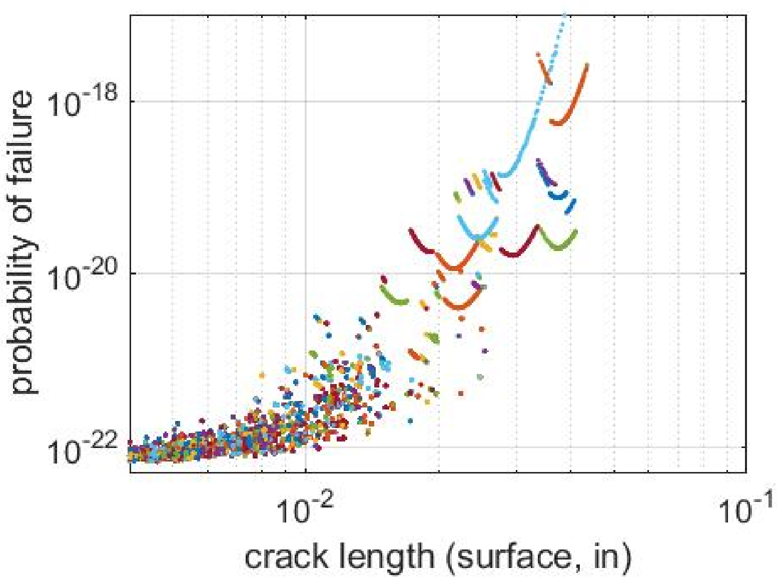


Figure 52: Crack trajectories in surface length and probability of failure.

The crack growth trajectories for the baseline forecast are shown in Figure 52. The plot shows that many cracks hardly grow at all. Cracks starting around .02 inches have moderate growth and a u-shaped trend is seen. The u-shaped curves indicate that as the surface dimension grows, the probability of failure initially decreases. While this may seem odd, it is in fact physical. During the initial phase of crack growth, the crack grows in the surface direction faster than the depth direction, causing a slightly elliptical crack. Once a stable crack aspect ratio is reached, the crack grows both in the depth and surface direction and the failure probability increases. This phenomenon can be seen in Figure 53. The aspect ratio initially decreases to a value of around 0.92 while the probability of failure decreases slightly. Thereafter the cracks stay around the same stable aspect ratio and grow, resulting in larger probability of failure. The phenomenon captured here is due to the detailed finite element modeling and is captured in the structure of the K_I model. In fact, one can calculate the stable crack aspect ratio (as a function of crack length) as the aspect ratio at which $K_{I,surf} = K_{I,depth}$. This indeed is around 0.92 for small cracks at the demonstration location. One could consider taking this idea a step further by initializing the aspect ratio to the stable aspect ratio.

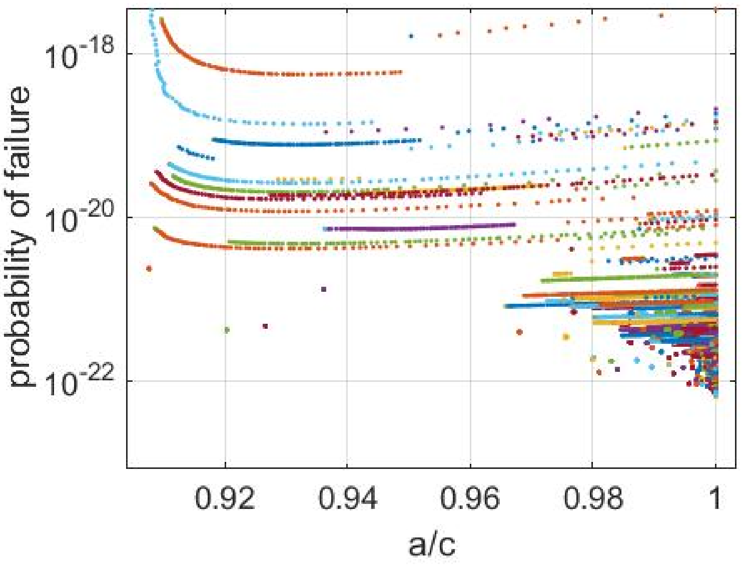
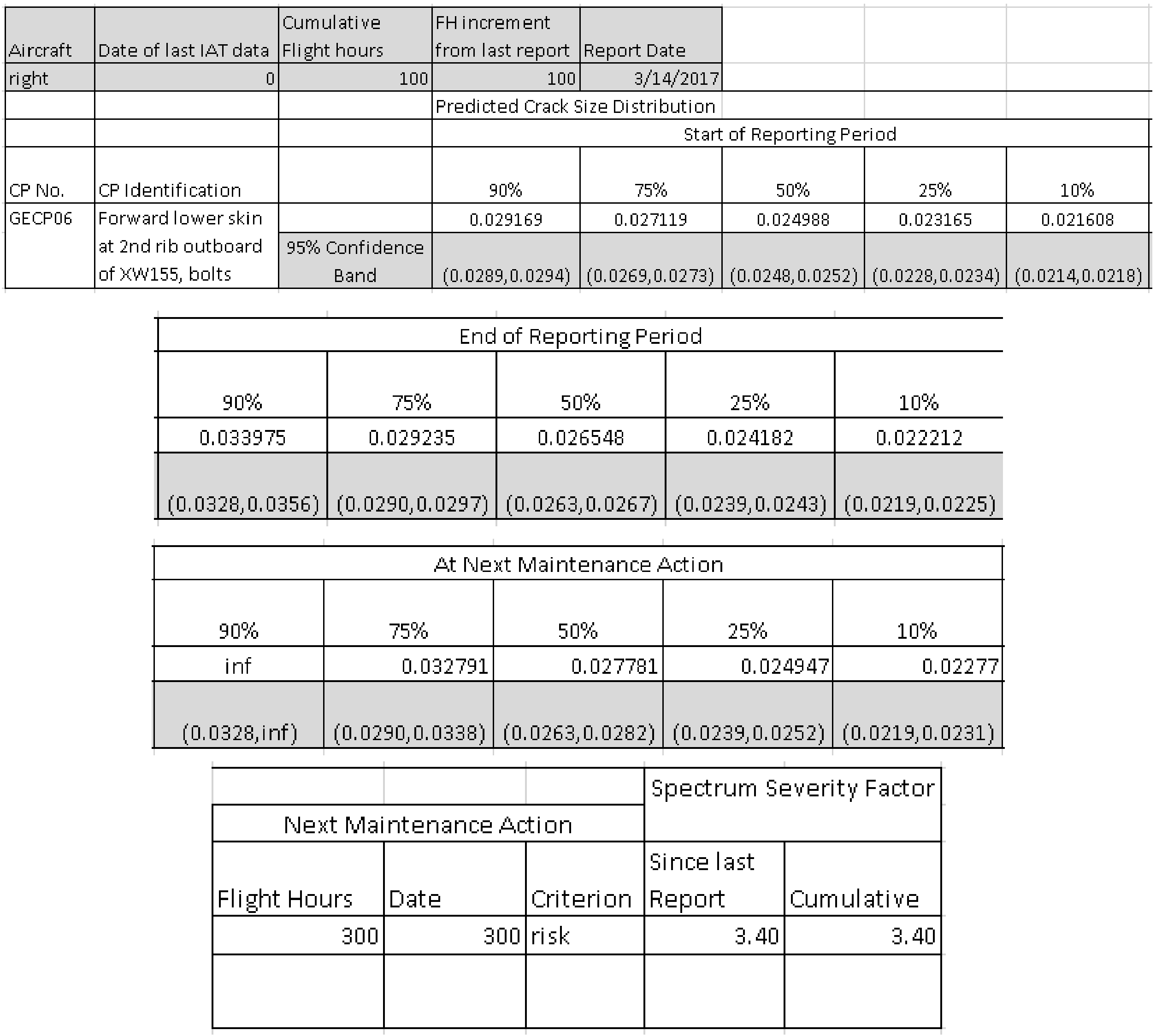


Figure 53: Crack trajectories in aspect ratio and probability of failure.



Aircraft	Date of last IAT data	Cumulative Flight hours	FH increment from last report	Report Date			
right	0	100	100	3/14/2017			
			Predicted Crack Size Distribution				
			Start of Reporting Period				
CP No.	CP Identification		90%	75%	50%	25%	10%
GECP06	Forward lower skin		0.029169	0.027119	0.024988	0.023165	0.021608
	at 2nd rib outboard of XW155, bolts	95% Confidence Band	(0.0289,0.0294)	(0.0269,0.0273)	(0.0248,0.0252)	(0.0228,0.0234)	(0.0214,0.0218)

End of Reporting Period				
90%	75%	50%	25%	10%
0.033975	0.029235	0.026548	0.024182	0.022212
(0.0328,0.0356)	(0.0290,0.0297)	(0.0263,0.0267)	(0.0239,0.0243)	(0.0219,0.0225)

At Next Maintenance Action				
90%	75%	50%	25%	10%
inf	0.032791	0.027781	0.024947	0.02277
(0.0328,inf)	(0.0290,0.0338)	(0.0263,0.0282)	(0.0239,0.0252)	(0.0219,0.0231)

			Spectrum Severity Factor	
Next Maintenance Action				
Flight Hours	Date	Criterion	Since last Report	Cumulative
300	300	risk	3.40	3.40

Figure 54: Control Point Summary Table Output for Data Assimilation Task

6.9 Data Assimilation

The next demonstration is assimilating 100 flights of data, performing updating with one inspection result, and subsequently forecasting with inspection scheduling. The IAT control point summary table output is shown in for the case in which the crack is detected (but not repaired) after 100 flights. The distribution at the start of the reporting period is the initial flaw size distribution. Note that the confidence bounds on the percentiles were computed via bootstrapping.

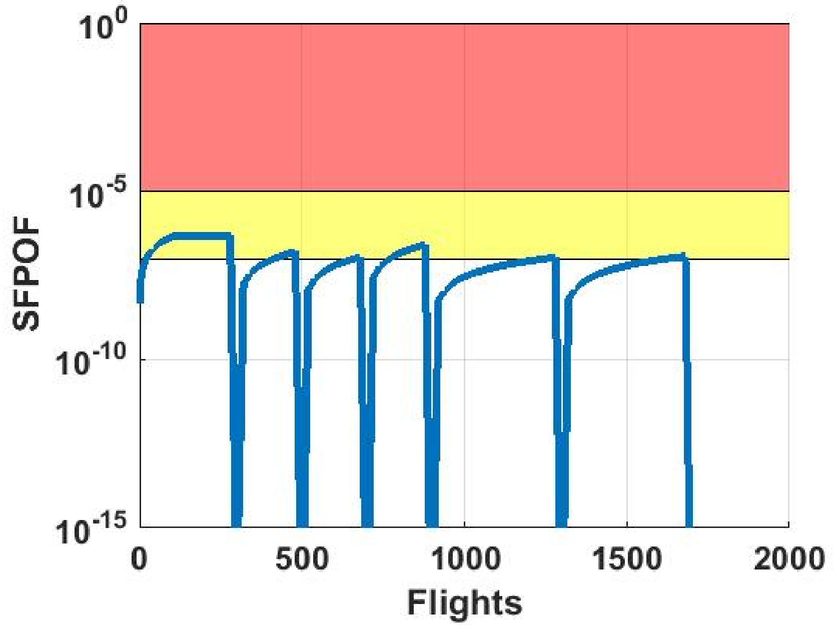


Figure 55: Probability of failure for the crack detected case.

The probability of failure forecast in the case in which the crack is detected (but not repaired) after 100 flights is shown in Figure 55. The computations result in inspections scheduled at 300, 500, 700, 900, 1300, and 1700 flights after the forecasting begins. The inspections are done because the probability of failure will exceed the threshold of 10^{-7} before the next potential inspection time. However, the probability of detecting a crack at each inspection is low (10-30%), so the tool assumes that most inspections result in cracks not found and thus not repaired. This makes sense because if there is a 1/10,000,000 chance that the crack is large enough to cause failure (SFPOF threshold), then there is also around a 1/10,000,000 chance that it is big enough to be found with the inspection method. The fact that the inspection method is viable down to cracks around .05 inches reduces the 1/10,000,000 number to more like 1/10, but still leaves a good chance that the crack will not be found or repaired. A large crack that misses detection will push up the failure probability quite high (e.g. up to 10^{-1}). This accounts for the potentially high probability of failure even shortly after inspections.

The forecast in the case in which the crack is not detected is shown in Figure 56. Inspections are scheduled for flights 300, 700, and 900. Enough information is learned during these inspections that subsequent inspections can be skipped. This shows the cost benefit of performing risk-based inspection scheduling and shows how P^2IAT can use the information that the crack was not detected to reduce the number of required inspections compared to the baseline.

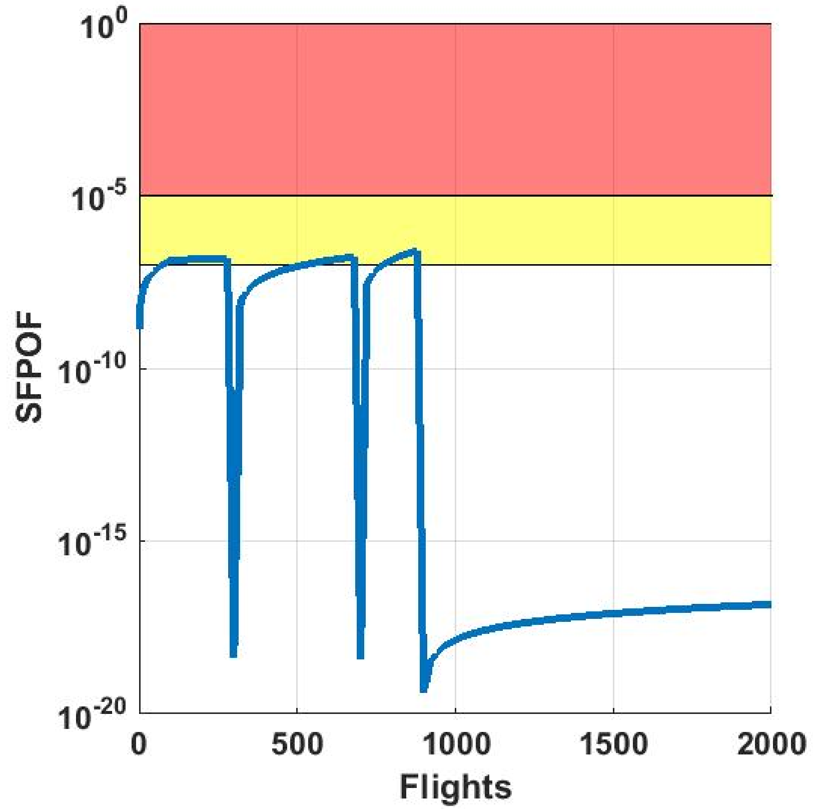


Figure 56: Probability of failure for the crack not detected case.

In this demonstration, all inspections were performed when the crack length was relatively low compared to the detection threshold. This is because the upper tail of the crack length distribution was large enough to pass the risk threshold of SFOF $> 10^{-7}$, while at the same time the majority of cracks were not detectable. Therefore, information-gain based inspection scheduling would not have had any different results. Indeed, the results imply that more information would be learned if one was willing to tolerate operating in higher SFPOF. Alternatively, more information would be learned if the inspection method had a lower detection limit.

The results of the P2IAT demonstration show that in the baseline, with no extra information, the tool calculates that all regular inspections (every 200 flights) should be performed. The tool can utilize both detect and no-detect information to reduce the number of required inspections. In the crack detected case uncertainty is reduced (lowering risk) due to the information gained but the crack is found to be large (raising risk). These competing effects result in 6 inspections being scheduled (instead of 5 in the baseline). In the crack not detected case uncertainty is reduced (lowering risk) and the crack is found to be small (lowering risk). This results in 3 inspections being scheduled and in general a lower SFPOF. In all cases, inspections become less frequent over time because the tool learns more about the aircraft.

7.0 CONCLUSIONS

A framework for Scalable, Accurate, Flexible, Efficient, and Robust Probabilistic and Prognostic Individual Aircraft Tracking has been developed and demonstrated. The framework leverages several advanced probabilistic methods to build meta-models, perform calibration, propagate uncertainty through various engineering analyses, update and reduce uncertainty with data, and forecast usage and damage into the future. The framework is flexible enough to apply to many types of assets and damage mechanisms.

An important result of this effort is linking a series of engineering analyses which are typically performed departmentally into a single, unified framework. These include:

- Usage analysis – missions, flight profile
- External loads analysis – vehicle level FEM
- Internal stress analysis – at particular control points
- Crack growth analysis
- Inspection method development and inspection results

By linking all of these analyses, the connections between them become more apparent. Indeed, the connections must be explicitly defined and mapped out, as in Figure 17. A second benefit is that uncertainty is quantified for each analysis in turn and propagated through to decision making criteria (e.g. failure probabilities). This allows a clear picture of which analyses are impacting decisions, which need more information or more detailed work, etc. A third benefit is that the framework allows data collection (e.g. inspection data) to be leveraged by all of the analyses. The Bayesian updating methods naturally target uncertain input parameters which are both causing much uncertainty and are most informed by whatever data are collected. Finally, the inspection scheduling methodology helps predict what data should be collected in the first place.

The result is a framework that can help those managing complex assets unify and develop their engineering analyses to provide machine-specific insights into operation and maintenance.

8. REFERENCES

[1] K. Q. Weinberger, J. Blitzer and L. K. Saul, "Distance metric learning for large margin nearest neighbor classification.," *Advances in neural information processing systems,* 2005.

[2] H.-S. Park and C.-H. Jun, "A simple and fast algorithm for K-medoids clustering," *Expert Systems with Applications,* vol. 36, pp. 3336-3341, 2009.

[3] I. Rychlik, "Simulation of load sequences from Rainflow matrices: Markov method.," *International Journal of Fatigue,* vol. 18, no. 7, pp. 429-438, 1996.

[4] J. D. Clothiaux and N. E. Dowling, "Verification of rain-flow reconstructions of a variable amplitude load history.," NASA Langley Research Center, 1992.

[5] K. Halbert and L. M. Fitzwater, "Sequential Bayesian Approach to Probabilistic Damage Tolerance Analysis.," in *55th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference, AIAA SciTech*, 2014.

[6] A. Doucet, N. De Freitas, K. Murphy and S. Russell, "Rao-Blackwellised particle filtering for dynamic Bayesian networks.," in *Proceedings of the Sixteenth conference on Uncertainty in artificial intelligence*, 2000.

[7] N. Friedman, K. Murphy and S. Russell, "Learning the structure of dynamic probabilistic networks," *Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence,* pp. 139-147, 1998.

[8] G. Bartram and S. Mahadevan, "Integration of heterogeneous information in SHM models," *Struct. Control Heal. Monit.,* vol. 21, no. 3, pp. 403-422, 2014.

[9] A. Doucet, S. Godsill and C. Andrieu, "On sequential Monte Carlo sampling methods for Bayesian filtering.," *Statistics and computing,* vol. 10, no. 3, pp. 197-208, 2000.

[10] B. W. Silverman, "Density Estimation for Statistics and Data Analysis," *Monographs on Statistics and Applied Probability.,* vol. 26, 1986.

[11] D. M. Blei, A. Y. Ng and M. I. Jordan, "Latent dirichlet allocation.," *Journal of machine Learning research,* vol. 3, pp. 993-1022, 2003.

[12] J. Oakley and A. O'Hagan, "Bayesian inference for the uncertainty distribution of computer model outputs," *Biometrika,* vol. 89, no. 4, pp. 769-784, 2002.

[13] M. C. Kennedy and A. O'Hagan, "Bayesian calibration of computer models.," *Journal of the Royal Statistical Society: Series B (Statistical Methodology),* vol. 63, no. 3, pp. 425-464, 2001.

[14] N. Pugno and e. al, "A Generalized Paris' Law for Fatigue Crack Growth," *Journal of the Mechanics and Physics of Solids,* vol. 54, no. 7, pp. 1333-1349, 2006.

[15] C. F. Babilon, R. H. Wygonik and G. E. Nordmark, "Mechanical Properties, Fracture Toughness, Fatigue, Environmental Fatigue Crack Growth Rates and Corrosion Characteristics of High-Toughness Aluminum Alloy Forgings, Sheet and Plate," Aluminum Company of America, 1973.

[16] J. C. Helton and F. J. Davis, "Latin hypercube sampling and the propagation of uncertainty in analyses of complex systems.," *Reliability Engineering & System Safety,* vol. 81, no. 1, pp. 23-69, 2003.

[17] D. L. Ball and M. T. Doerfler, "Stress Intensity Factor Solution Development for Interference Fit and Cold Expanded Holes," Lockheed Martin Corporation, 1999.

[18] H. Tada, P. C. Paris and G. R. Irwin, The stress analysis of cracks., Hellertown, PA: Del Research Corp, 1973.

[19] ASTM International, "ASTM E399-12e3 Standard Test Method for Linear-Elastic Plane-Strain Fracture Toughness KIc of Metallic Materials.," ASTM International, West Conshohocken, PA, 2012.

[20] D. Skinn, J. P. Gallagher, A. P. Berens, P. Huber and J. Smith, "USAF Damage Tolerant Design Handbook," U.S. Air Force Research Laboratory, Wright-Patterson Air Force Base, OH, 1994.

[21] J. Doke, "7173-grabit," 01 02 2015. [Online]. Available: https://www.mathworks.com/matlabcentral/fileexchange/7173-grabit.

[22] ASTM International, "ASTM E1049-85(2011)e1 Standard Practices for Cycle Counting in Fatigue Analysis.," ASTM International, West Conshohocken, PA, 2011.

[23] P. C. Miedlar, A. P. Berens, A. Gunderson and J. P. Gallagher, "USAF Damage Tolerant Design Handbook: Guidelines for the Analysis and Design of Damage Tolerant Aircraft Structure," U.S. Air Force Research Laboratory, Wright-Patterson Air Force Base, OH, 2002.

[24] C. Annis, "Nondestructive Evaluation System Reliability Assessment.," Wright-Patterson AFB, USA, 2009.

[25] Department of Defense, "Nondestrctive Evaluation System Reliability Assessment," United States Department of Defense, Washington, DC, 2009.

[26] W. Chen, R. Jin and A. Sudjianto, "Analytical variance-based global sensitivity analysis in simulation-based design under uncertainty.," *Journal of mechanical design,* vol. 127, no. 5, pp. 875-886, 2005.

[27] J. Lin, "Divergence Measures Based on the Shannon Entropy.," *IEEE Transactions on Information Theory,* vol. 37, no. 1, pp. 145-51, 1999.

[28] The HDF Group, "Hierarchical Data Format, version 5," 1997-2016.

[29] D. J. White and T. D. Gray, "Damage Tolerance Assessment of the A-7D Aircraft Structure," in *ICF5*, Cannes (France), 1981.

[30] E. A. Wan, R. Van Der Merwe and A. T. Nelson, "Dual Estimation and the Unscented Transformation.," *NIPS,* 1999.

[31] P. Noronha and et.al., "Fastener Hole Quality," AFFDL-TR-78-206, Vol. 1, 1978.

[32] K. P. Murphy, "Dynamic Bayesian Networks: Representation, Inference and Learning," 2002.

[33] A. Makeev and et.al., "A Concept for Quantifying Equivalent Initial Flaw Size Distribution Fracture Mechanics Based Life Prediction Models," *International Jounral of Fatigue,* p. 29, 2007.

[34] P. Magnusen and et.al., "The Role of Microstructure in the Fatigue Durability of Aluminum Aircraft Alloys," ONR Contract No. N00014-91-C-0128, 1995.

[35] A. Haldar and S. Mahadeavn, Probability, Reliability, and Statistical Methods in Engineering Design., John Wiley & Sons, 2000.

[36] S. Fawaz, "Equivalent Initial Flaw Size Testing and Analysis of Transport Aircraft Skin Splices," *Fatigue and Fracture of Engineering Materials and Structures,* p. 26, 2003.

[37] D. Dawicke, F. Brust, I. Raju and D. Cheston, "Residual Stresses and Critical Initial Flaw Size Analyses of Welds," in *50th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, 2009.

[38] J. Bourbonnais, A. Dietz, D. Pitt, E. Reichenbach, E. Ritz and P. C. Chen, "Air Vehicle Integration and Technology Research: Stick-to-Stress Real-Time Simulator Application," Air Force Research Laboratory, Wright-Patterson Air Force Base, 2014.

[39] E. J. Antolik, "Stress Analysis of F-15 Wing," McDonnel Aircraft Company, Saint Louis, 1972.

[40] GE Global Research, "White Paper on Initial Flaw Size Distribution," GE, Niskayuna, 2015.

[41] P. Paris, R. McMeeking and H. Tada, "The Weight Function Method for determining Stress Intensity Factors," *Cracks and Fracture, ASTM STP 601, American Society for Testing Materials,* pp. 471-489, 1976.

[42] S. Lauritzen, "Propagation of Probabilities, Means and Variances in Mixed Graphical Association Models," *J. Am. Stat. Assoc.,* vol. 87, pp. 1098-1108, 1992.

[43] F. Pedregosa and e. al., "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research,* vol. 12, pp. 2825-2830, 2011.

Appendix A – 6DOF Flight Simulator

The following pages include a listing of the 110 runs of the 6DOF flight simulator (Stick-to-Stress V1.0) used to build the flight recorder data to external load model and the baseline usage data (MES). The input to the simulator was one of 10 ICFlags (initial conditions) and a piecewise continuous time history of control stick movements (longitudinal and lateral stick forces). For each maneuver, the simulator produced 3 to 10 seconds of time history flight recorder and loads data.

Each maneuver is identified by its maneuver type (Symmetric pull up/push over, Rolling Pull Out (RPO), 360 degree roll, or negative 1g 180 degree roll), ICFlag (initial condition flag, 1-10), and stick forces. The time from the start of the maneuver followed by the stick force are given in the columns t1, f1, t2, f2, etc. These columns describe a piecewise linear time-varying stick force, f(t), in lbs. For symmetric maneuvers, this is the longitudinal stick force. For the other asymmetric maneuvers, this is the lateral stick force.

For symmetric maneuvers, the lateral stick force is zero always (pure longitudinal force). For asymmetric maneuvers, the longitudinal stick force is zero always (pure lateral force).

Table A-1. Symmetric Maneuver Inputs to 6DOF Flight Simulator.

Maneuver Type	ID	MACH	ALT	ICFlag	Max NNZCG	t1	f1	t2	f2	t3	f3	t4	f4	t5	f5
Symmetric	1	0.7	500	1	8.44	0	0	0.2	50	0.585	50	0.785	0	2	0
Symmetric	2	0.7	1500	2	8.43	0	0	0.2	50	0.595	50	0.795	0	2	0
Symmetric	3	0.8	10000	3	8.43	0	0	0.2	50	0.592	50	0.792	0	2	0
Symmetric	4	0.82	20000	4	7.79	0	0	0.2	50	0.827	50	1.027	0	2	0
Symmetric	5	0.9	10000	5	8.40	0	0	0.2	50	0.427	50	0.627	0	2	0
Symmetric	6	0.9	20000	6	8.44	0	0	0.2	50	0.745	50	0.945	0	2	0
Symmetric	7	0.95	20000	7	8.41	0	0	0.2	50	0.615	50	0.815	0	2	0
Symmetric	8	1.04	10000	8	8.45	0	0	0.2	50	0.208	50	0.408	0	2	0
Symmetric	9	1.05	20000	9	8.45	0	0	0.2	45	0.2	45	0.4	0	2	0
Symmetric	10	1.15	10000	10	8.51	0	0	0.2	30.1	0.2	30.1	0.4	0	2	0
Symmetric	11	0.7	500	1	6.17	0	0	0.2	50	0.225	50	0.425	0	2	0
Symmetric	12	0.7	1500	2	6.16	0	0	0.2	50	0.228	50	0.428	0	2	0
Symmetric	13	0.8	10000	3	6.15	0	0	0.2	50	0.212	50	0.412	0	2	0
Symmetric	14	0.82	20000	4	6.08	0	0	0.2	50	0.246	50	0.446	0	2	0
Symmetric	15	0.9	10000	5	6.19	0	0	0.2	45.45	0.2	45.45	0.4	0	2	0
Symmetric	16	0.9	20000	6	6.06	0	0	0.2	48	0.2	48	0.4	0	2	0
Symmetric	17	0.95	20000	7	6.01	0	0	0.2	44.1	0.2	44.1	0.4	0	2	0
Symmetric	18	1.04	10000	8	6.02	0	0	0.2	28.29	0.2	28.29	0.4	0	2	0
Symmetric	19	1.05	20000	9	6.06	0	0	0.2	28.4	0.2	28.4	0.4	0	2	0
Symmetric	20	1.15	10000	10	5.58	0	0	0.2	22.32	0.2	22.32	0.4	0	2	0
Symmetric	21	0.7	500	1	4.08	0	0	0.2	36.44	0.2	36.44	0.4	0	2	0
Symmetric	22	0.7	1500	2	4.08	0	0	0.2	36.64	0.2	36.64	0.4	0	2	0
Symmetric	23	0.8	10000	3	4.06	0	0	0.2	35.9	0.2	35.9	0.4	0	2	0
Symmetric	24	0.82	20000	4	4.05	0	0	0.2	37.9	0.2	37.9	0.4	0	2	0
Symmetric	25	0.9	10000	5	4.02	0	0	0.2	30.02	0.2	30.02	0.4	0	2	0
Symmetric	26	0.9	20000	6	4.07	0	0	0.2	33.55	0.2	33.55	0.4	0	2	0
Symmetric	27	0.95	20000	7	3.99	0	0	0.2	28.24	0.2	28.24	0.4	0	2	0
Symmetric	28	1.04	10000	8	4.00	0	0	0.2	20.85	0.2	20.85	0.4	0	2	0
Symmetric	29	1.05	20000	9	4.03	0	0	0.2	20.06	0.2	20.06	0.4	0	2	0
Symmetric	30	1.15	10000	10	3.99	0	0	0.2	17.8	0.2	17.8	0.4	0	2	0
Symmetric	31	0.7	500	1	2.00	0	0	0.2	15.8	0.2	15.8	0.4	0	2	0
Symmetric	32	0.7	1500	2	2.00	0	0	0.2	15.7	0.2	15.7	0.4	0	2	0
Symmetric	33	0.8	10000	3	2.00	0	0	0.2	15.5	0.2	15.5	0.4	0	2	0
Symmetric	34	0.82	20000	4	1.99	0	0	0.2	16.9	0.2	16.9	0.4	0	2	0
Symmetric	35	0.9	10000	5	1.99	0	0	0.2	13.63	0.2	13.63	0.4	0	2	0
Symmetric	36	0.9	20000	6	2.00	0	0	0.2	15.3	0.2	15.3	0.4	0	2	0
Symmetric	37	0.95	20000	7	1.99	0	0	0.2	13.1	0.2	13.1	0.4	0	2	0
Symmetric	38	1.04	10000	8	1.99	0	0	0.2	10.15	0.2	10.15	0.4	0	2	0
Symmetric	39	1.05	20000	9	1.99	0	0	0.2	10.1	0.2	10.1	0.4	0	2	0
Symmetric	40	1.15	10000	10	1.97	0	0	0.2	8.7	0.2	8.7	0.4	0	2	0
Symmetric	41	0.7	500	1	-2.02	0	0	0.2	-38.9	0.2	-38.9	0.4	0	2	0
Symmetric	42	0.7	1500	2	-2.02	0	0	0.2	-39.6	0.2	-39.6	0.4	0	2	0
Symmetric	43	0.8	10000	3	-2.01	0	0	0.2	-37.6	0.2	-37.6	0.4	0	2	0
Symmetric	44	0.82	20000	4	-2.01	0	0	0.2	-42	0.2	-42	0.4	0	2	0
Symmetric	45	0.9	10000	5	-2.00	0	0	0.2	-32.3	0.2	-32.3	0.4	0	2	0
Symmetric	46	0.9	20000	6	-2.00	0	0	0.2	-36.5	0.2	-36.5	0.4	0	2	0
Symmetric	47	0.95	20000	7	-2.02	0	0	0.2	-31.9	0.2	-31.9	0.4	0	2	0
Symmetric	48	1.04	10000	8	-1.97	0	0	0.2	-22.6	0.2	-22.6	0.4	0	2	0
Symmetric	49	1.05	20000	9	-1.99	0	0	0.2	-22.8	0.2	-22.8	0.4	0	2	0
Symmetric	50	1.15	10000	10	-1.94	0	0	0.2	- 18.55	0.2	- 18.55	0.4	0	2	0

Table A-2. Rolling Pull-out Maneuver Inputs to 6DOF Flight Simulator.

Maneuver Type	ID	MACH	ALT	ICFlag	Max NNZCG	t1	f1	t2	f2	t3	f3	t4	f4	t5	f5	t6	f6
RPO	**1**	0.7	500	**1**	**6.46**	0	0	0.1	17.5	0.989	17.5	1.189	-17.5	1.279	-17.5	1.379	0
RPO	**2**	0.7	1500	**2**	**6.45**	0	0	0.1	17.5	0.991	17.5	1.191	-17.5	1.281	-17.5	1.381	0
RPO	**3**	0.8	10000	**3**	**6.59**	0	0	0.1	17.5	0.96	17.5	1.16	-17.5	1.25	-17.5	1.35	0
RPO	**4**	0.82	20000	**4**	**6.05**	0	0	0.1	17.5	1.166	17.5	1.366	-17.5	1.456	-17.5	1.556	0
RPO	**5**	0.9	10000	**5**	**6.72**	0	0	0.1	17.5	1.021	17.5	1.221	-17.5	1.311	-17.5	1.411	0
RPO	**6**	0.9	20000	**6**	**6.39**	0	0	0.1	17.5	0.965	17.5	1.165	-17.5	1.255	-17.5	1.355	0
RPO	**7**	0.95	20000	**7**	**6.70**	0	0	0.1	17.5	0.98	17.5	1.18	-17.5	1.27	-17.5	1.37	0
RPO	**8**	1.04	10000	**8**	**6.29**	0	0	0.1	17.5	1.366	17.5	1.566	-17.5	1.656	-17.5	1.756	0
RPO	**9**	1.05	20000	**9**	**6.22**	0	0	0.1	17.5	1.319	17.5	1.519	-17.5	1.609	-17.5	1.709	0
RPO	**10**	1.15	10000	**10**	**6.20**	0	0	0.1	17.5	1.699	17.5	1.899	-17.5	1.989	-17.5	2.089	0
RPO	**11**	0.7	500	**1**	**4.57**	0	0	0.954	17.5	1.154	-17.5	1.244	-17.5	1.344	0	1.379	0
RPO	**12**	0.7	1500	**2**	**4.56**	0	0	0.951	17.5	1.151	-17.5	1.241	-17.5	1.341	0	1.381	0
RPO	**13**	0.8	10000	**3**	**4.55**	0	0	0.93	17.5	1.13	-17.5	1.22	-17.5	1.32	0	1.35	0
RPO	**14**	0.82	20000	**4**	**4.69**	0	0	0.952	17.5	1.152	-17.5	1.242	-17.5	1.342	0	1.556	0
RPO	**15**	0.9	10000	**5**	**4.73**	0	0	1.024	17.5	1.224	-17.5	1.314	-17.5	1.414	0	2	0
RPO	**16**	0.9	20000	**6**	**4.64**	0	0	0.952	17.5	1.152	-17.5	1.242	-17.5	1.342	0	1.355	0
RPO	**17**	0.95	20000	**7**	**4.85**	0	0	1.003	17.5	1.203	-17.5	1.293	-17.5	1.393	0	2	0
RPO	**18**	1.04	10000	**8**	**4.58**	0	0	1.336	17.5	1.536	-17.5	1.626	-17.5	1.726	0	1.756	0
RPO	**19**	1.05	20000	**9**	**4.59**	0	0	1.27	17.5	1.47	-17.5	1.56	-17.5	1.66	0	1.709	0
RPO	**20**	1.15	10000	**10**	**4.63**	0	0	1.701	17.5	1.901	-17.5	1.991	-17.5	2.091	0	2.5	0
RPO	**21**	0.7	500	**1**	**3.06**	0	0	0.903	17.5	1.103	-17.5	1.193	-17.5	1.293	0	1.379	0
RPO	**22**	0.7	1500	**2**	**3.07**	0	0	0.896	17.5	1.096	-17.5	1.186	-17.5	1.286	0	1.381	0
RPO	**23**	0.8	10000	**3**	**3.06**	0	0	0.894	17.5	1.094	-17.5	1.184	-17.5	1.284	0	1.35	0
RPO	**24**	0.82	20000	**4**	**3.06**	0	0	0.909	17.5	1.109	-17.5	1.199	-17.5	1.299	0	1.556	0
RPO	**25**	0.9	10000	**5**	**3.18**	0	0	1.011	17.5	1.211	-17.5	1.301	-17.5	1.401	0	1.411	0
RPO	**26**	0.9	20000	**6**	**3.23**	0	0	0.929	17.5	1.129	-17.5	1.219	-17.5	1.319	0	1.355	0
RPO	**27**	0.95	20000	**7**	**3.10**	0	0	1.036	17.5	1.236	-17.5	1.326	-17.5	1.426	0	2	0
RPO	**28**	1.04	10000	**8**	**3.08**	0	0	1.289	17.5	1.489	-17.5	1.579	-17.5	1.679	0	1.756	0
RPO	**29**	1.05	20000	**9**	**3.07**	0	0	1.191	17.5	1.391	-17.5	1.481	-17.5	1.581	0	1.709	0
RPO	**30**	1.15	10000	**10**	**3.23**	0	0	1.627	17.5	1.827	-17.5	1.917	-17.5	2.017	0	2.089	0
RPO	**31**	0.7	500	**1**	**1.69**	0	0	0.769	17.5	0.969	-17.5	1.059	-17.5	1.159	0	1.379	0
RPO	**32**	0.7	1500	**2**	**1.70**	0	0	0.762	17.5	0.962	-17.5	1.052	-17.5	1.152	0	1.381	0
RPO	**33**	0.8	10000	**3**	**1.70**	0	0	0.767	17.5	0.967	-17.5	1.057	-17.5	1.157	0	1.35	0
RPO	**34**	0.82	20000	**4**	**1.72**	0	0	0.764	17.5	0.964	-17.5	1.054	-17.5	1.154	0	1.556	0
RPO	**35**	0.9	10000	**5**	**1.74**	0	0	0.882	17.5	1.082	-17.5	1.172	-17.5	1.272	0	1.411	0
RPO	**36**	0.9	20000	**6**	**1.73**	0	0	0.801	17.5	1.001	-17.5	1.091	-17.5	1.191	0	1.355	0

Table A-3. Rolling Pull-out Maneuver Inputs to 6DOF Flight Simulator. (continued)

Maneuver Type	ID	MACH	ALT	ICFlag	Max NNZCG	t1	f1	t2	f2	t3	f3	t4	f4	t5	f5	t6	f6
RPO	**37**	0.95	20000	**7**	**1.64**	0	0	0.887	17.5	1.087	-17.5	1.177	-17.5	1.277	0	1.37	0
RPO	**38**	1.04	10000	**8**	**1.66**	0	0	1.115	17.5	1.315	-17.5	1.405	-17.5	1.505	0	1.756	0
RPO	**39**	1.05	20000	**9**	**1.60**	0	0	1.017	17.5	1.217	-17.5	1.307	-17.5	1.407	0	1.709	0
RPO	**40**	1.15	10000	**10**	**1.87**	0	0	1.373	17.5	1.573	-17.5	1.663	-17.5	1.763	0	2.089	0

Table A-4. Roll Maneuver Inputs to 6DOF Flight Simulator.

Maneuver Type	ID	MACH	ALT	ICFlag	Max NNZCG	t1	f1	t2	f2	t3	f3	t4	f4	t5	f5	t6	f6	t7	f7	t8	f8	t9	f9	t10	f10
Roll_360	**1**	0.9	10000	**1**	**1.56**	0	0	0.1	17.5	1.716	17.5	1.916	-17.21	1.929	-17.5	2.054	-17.5	2.066	-16.4	2.154	-1.084	2.166	0	10	0
Roll_360	**2**	0.9	10000	**2**	**1.56**	0	0	0.1	17.5	1.696	17.5	1.896	-17.21	1.909	-17.5	2.034	-17.5	2.046	-16.4	2.134	-1.084	2.146	0	10	0
Roll_360	**3**	0.9	10000	**3**	**1.56**	0	0	0.1	17.5	1.717	17.5	1.917	-17.21	1.929	-17.5	2.054	-17.5	2.067	-16.4	2.154	-1.084	2.167	0	10	0
Roll_360	**4**	0.9	10000	**4**	**1.56**	0	0	0.1	17.5	1.717	17.5	1.917	-17.21	1.93	-17.5	2.055	-17.5	2.067	-16.4	2.155	-1.084	2.167	0	10	0
Roll_360	**5**	0.9	10000	**5**	**1.56**	0	0	0.1	17.5	2.013	17.5	2.213	-17.21	2.225	-17.5	2.35	-17.5	2.363	-16.4	2.45	-1.084	2.463	0	10	0
Roll_360	**6**	0.9	10000	**6**	**1.56**	0	0	0.1	17.5	1.806	17.5	2.006	-17.21	2.018	-17.5	2.143	-17.5	2.156	-16.4	2.243	-1.084	2.256	0	10	0
Roll_360	**7**	0.9	10000	**7**	**1.56**	0	0	0.1	17.5	1.983	17.5	2.183	-17.21	2.195	-17.5	2.32	-17.5	2.333	-16.4	2.42	-1.084	2.433	0	10	0
Roll_360	**8**	0.9	10000	**8**	**1.63**	0	0	0.1	17.5	4.247	17.5	4.447	-17.21	4.46	-17.5	4.585	-17.5	4.597	-16.4	4.685	-1.084	4.697	0	10	0
Roll_360	**9**	0.9	10000	**9**	**1.62**	0	0	0.1	17.5	2.284	17.5	2.484	-17.21	2.496	-17.5	2.621	-17.5	2.634	-16.4	2.721	-1.084	2.734	0	10	0
Roll_360	**10**	0.9	10000	**10**	**1.63**	0	0	0.1	17.5	4.814	17.5	5.014	-17.21	5.027	-17.5	5.152	-17.5	5.164	-16.4	5.252	-1.084	5.264	0	10	0
Roll_180_ng	**1**	0.9	10000	**1**	**-1.59**	0	0	0.1	17.5	0.912	17.5	1.112	-17.15	1.124	-17.5	1.287	-17.5	1.387	-0.314	1.399	0	10	0		
Roll_180_ng	**2**	0.9	10000	**2**	**-1.59**	0	0	0.1	17.5	0.912	17.5	1.112	-17.15	1.124	-17.5	1.287	-17.5	1.387	-0.314	1.399	0	10	0		
Roll_180_ng	**3**	0.9	10000	**3**	**-1.60**	0	0	0.1	17.5	0.916	17.5	1.116	-17.15	1.128	-17.5	1.291	-17.5	1.391	-0.314	1.403	0	10	0		
Roll_180_ng	**4**	0.9	10000	**4**	**-1.57**	0	0	0.1	17.5	0.884	17.5	1.084	-17.15	1.097	-17.5	1.259	-17.5	1.359	-0.314	1.372	0	10	0		
Roll_180_ng	**5**	0.9	10000	**5**	**-1.72**	0	0	0.1	17.5	1.225	17.5	1.425	-17.15	1.438	-17.5	1.6	-17.5	1.7	-0.314	1.713	0	10	0		
Roll_180_ng	**6**	0.9	10000	**6**	**-1.64**	0	0	0.1	17.5	0.978	17.5	1.178	-17.15	1.19	-17.5	1.353	-17.5	1.453	-0.314	1.465	0	10	0		
Roll_180_ng	**7**	0.9	10000	**7**	**-1.71**	0	0	0.1	17.5	1.169	17.5	1.369	-17.15	1.382	-17.5	1.544	-17.5	1.644	-0.314	1.657	0	10	0		
Roll_180_ng	**8**	0.9	10000	**8**	**-1.35**	0	0	0.1	17.5	2.219	17.5	2.419	-17.15	2.432	-17.5	2.594	-17.5	2.694	-0.314	2.707	0	10	0		
Roll_180_ng	**9**	0.9	10000	**9**	**-1.56**	0	0	0.1	17.5	1.639	17.5	1.839	-17.15	1.851	-17.5	2.014	-17.5	2.114	-0.314	2.126	0	10	0		
Roll_180_ng	**10**	0.9	10000	**10**	**-1.00**	0	0	0.1	17.5	2.927	17.5	3.127	-17.15	3.139	-17.5	3.302	-17.5	3.402	-0.314	3.414	0	10	0		

Appendix B – Verification of Rainflow Cycle Counting

This section describes verification work of the python implementations of rainflow cycle counting and truncation. Although truncation is not used in the final P^2IAT deliverable per the Red-Beard Team review, it is included here for completeness.

A previous study on aircraft flight records obtained from the Air Force showed that many of the load cycles had high R-ratios. The R-ratio is the ratio of minimum to maximum stress in a cycle:

$$R = \frac{\sigma_{min}}{\sigma_{max}} \tag{B-1}$$

High R-ratio cycles are cycles in which the min and max stress are both large; i.e. the stress history oscillates around a large positive value. It was suggested that real flight records should not have so many high R-ratio cycles, and therefore the cycle counting algorithms may be incorrectly implemented.

First, the cycle content produced by the algorithms was analyzed for consistency. The original presentation of the cycles was a histogram of R-ratios, shown in Figure B-1, which points out the high R-ratio content. A scatter plot of mean stress and R-ratio was created, shown in Figure B-2. The plot shows, as expected, that many high R-ratio cycles exist. In addition, those cycles occur at high mean stress (oscillations around a large stress) but also at low mean stress. Cycles with high R-ratio but low mean stress are very small oscillations about a low stress value. These cycles are likely to be more important to crack growth than the high mean stress cycles. Thus, filtering cycles based on R-ratio alone would adversely affect the accuracy of crack growth calculations.

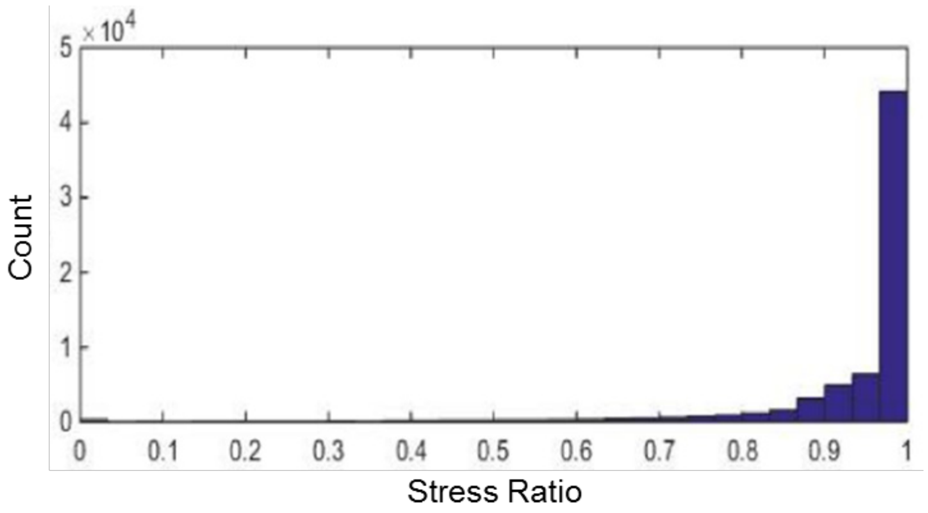


Figure B-1. Histogram of Stress Ratios for Example Flight Records.

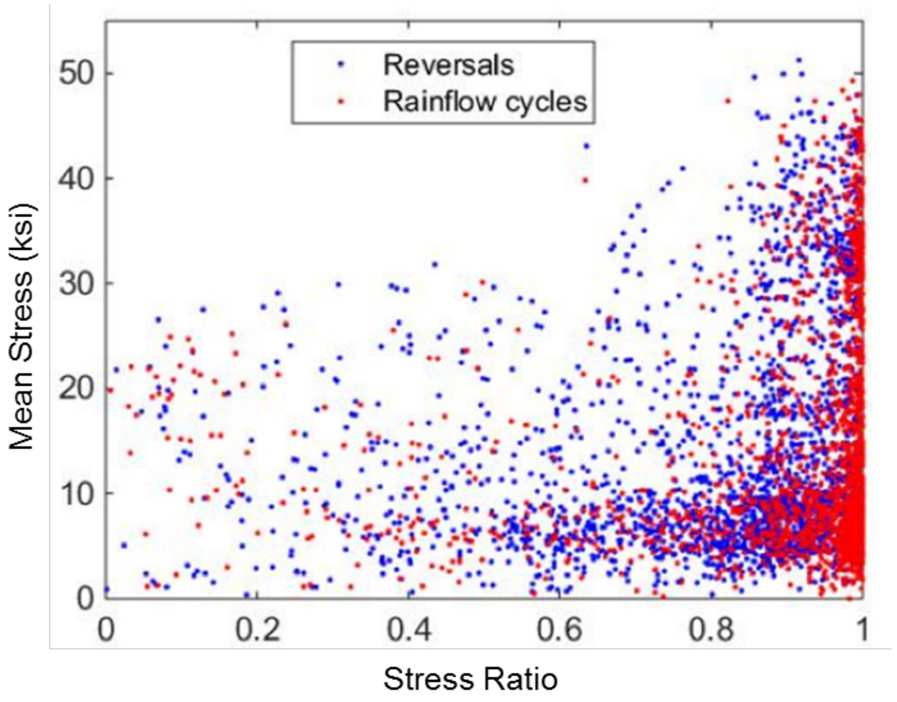


Figure B-2. Mean Stress and R-ratio for Cycles from an Example Flight Record.

Figure B-2 also shows the result of counting each reversal as a cycle. While not accurate for crack growth, this method has the advantage of being easier to interpret, since each cycle has a known time point at which it occurs. In contrast, rainflow counted cycles have a particular order, but no associated time point. The reversal method gives similar cycle content to rainflow counting and was used for debugging purposes.

After debugging, the algorithms were verified. The GE team and Lockheed Martin team each ran full cycle counting algorithms on the same data set and compared the results. The Lockheed Martin team used their standard process, which was assumed to be correct. The cycle counting algorithm consists of three parts: extrema extraction, truncation, and rainflow counting. First, the results without any truncation were compared, since it was previously recommended that truncation be turned off. The results are shown in Figure B-3. The plot shows that both algorithms give the same result, thus verifying the GE implementation. Note that there are indeed many cycles with high R-ratio. The plot looks less extreme than Figure B-1 only because the y-axis is on a log-scale.

Next, the same comparison was run with truncation set to 1% of the maximum stress value. It is standard procedure for Lockheed Martin to set the truncation level to 1%-5% which increases computational speed without compromising accuracy. Figure B-4 shows that the results are again the same. Note the decrease in the number of high R-ratio cycles. Still, though, the histogram peaks at a relatively high R-ratio of 0.9.

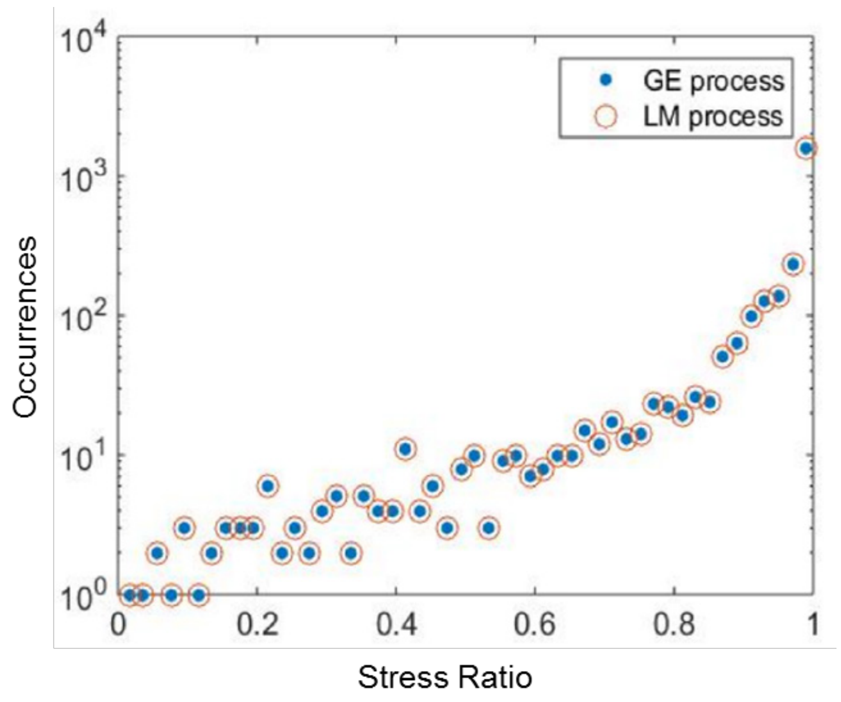


Figure B-3. Histogram of R-Ratios Comparing of GE and Lockheed Martin Cycle Counting Processes without Truncation.

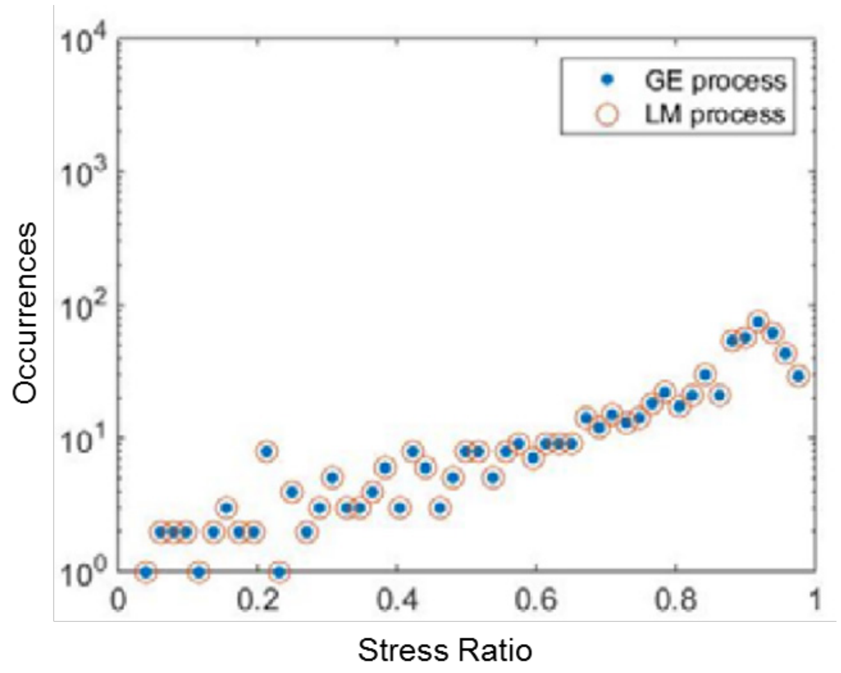


Figure B-4. Histogram of R-Ratios Comparing GE and Lockheed Martin Processes with Truncation at 1% of the Maximum Stress.

After the GE algorithms were verified, the question still remains as to the origin of the high R-ratio content in the flight records. As a comparison, a number of synthetic "flights" were created by concatenating maneuver loads calculated by Stick-to-Stress V1.0 (StS). It was postulated that the high R-ratio content may exist for real flight data but not simulated load cases. With truncation applied, the histograms of 20 synthetic flights are shown in Figure B-5. The plots show that the StS maneuvers include more low R-ratio content than occurred in the example flights shown in Figure B-1. Thus, it is possible that the suggestions that one should see low R-ratio content are related to design load cases, whereas for real flight data it is indeed reasonable to see more high R-ratio content.

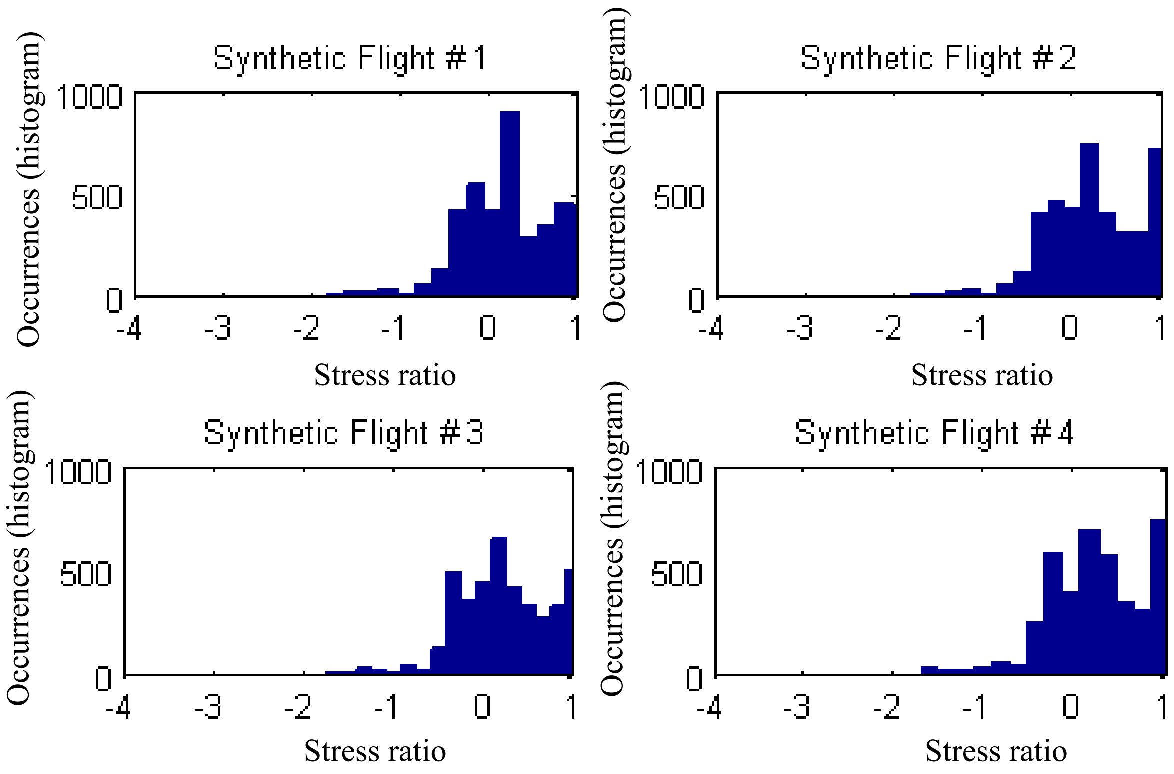


Figure B-5. Histograms of R-Ratios for Synthetic Flights Created from Stick-to-Stress Output.

Finally, the effect of truncation on crack growth was assessed. Figure B-6 shows deterministic crack growth curves without truncation and with 1% truncation. As expected, the two are nearly identical.

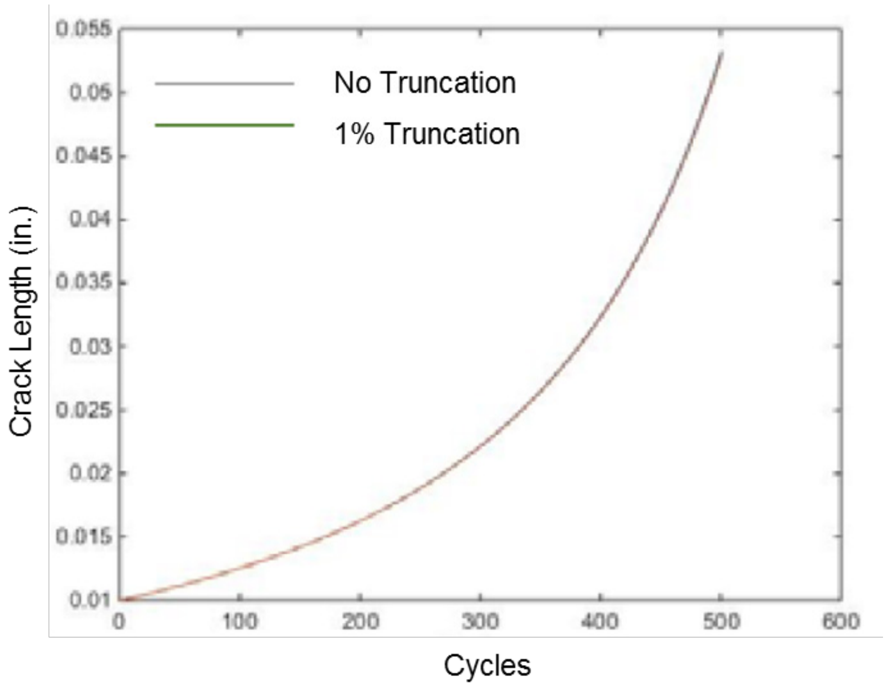


Figure B-6. Effect of Truncation on Crack Growth Predictions.

The GE implementation of cycle counting algorithms has been verified to match the standard Lockheed Martin process. In addition, it was found that the F-15C flight records do indeed contain much high R-ratio content, while StS load cases contain less. Much of the high R-ratio content in real flight data can be filtered out by truncation, which reduces computational time without compromising accuracy in crack growth. Still, due to Red-Beard Team suggestions, truncation will be turned off in the final deliverable of P^{2}IAT.

Appendix C – Comparison of MES to other aircraft

The Lockheed team's analysis of the original MES indicated that it is very damaging. Comparison with spectra from other tactical aircraft showed that one possible reason for this is that there is more negative bending content in the MES for this test than for those aircraft, see Figure C-1. The probabilistic load spectrum was modified so that maneuvers with high negative bending content had lower probability of occurring. The probability scales with the minimum bending moment in the maneuver. In addition, maneuvers with a normalized negative bending moment of less than -0.3 were removed (this constituted just 3 of the 110 maneuvers). The MES was re-generated using the new maneuver probabilities and the bending moment exceedances are shown in Figure C-2.

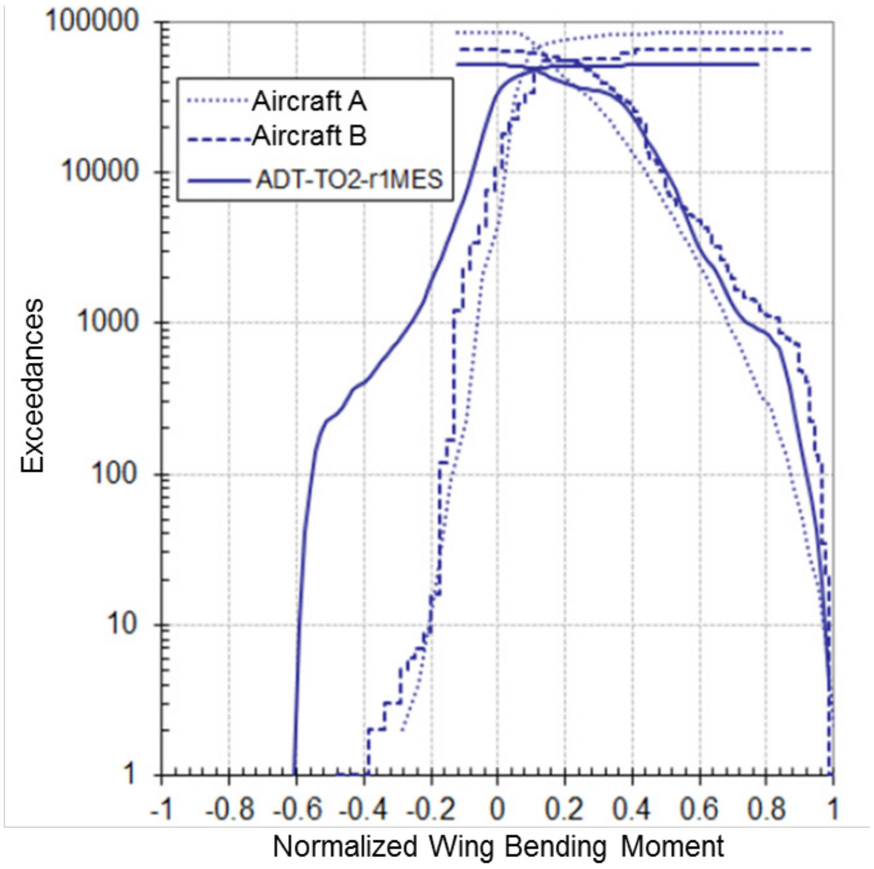


Figure C-1. Comparison of Baseline MES Exceedances to Typical Tactical Aircraft Exceedances.

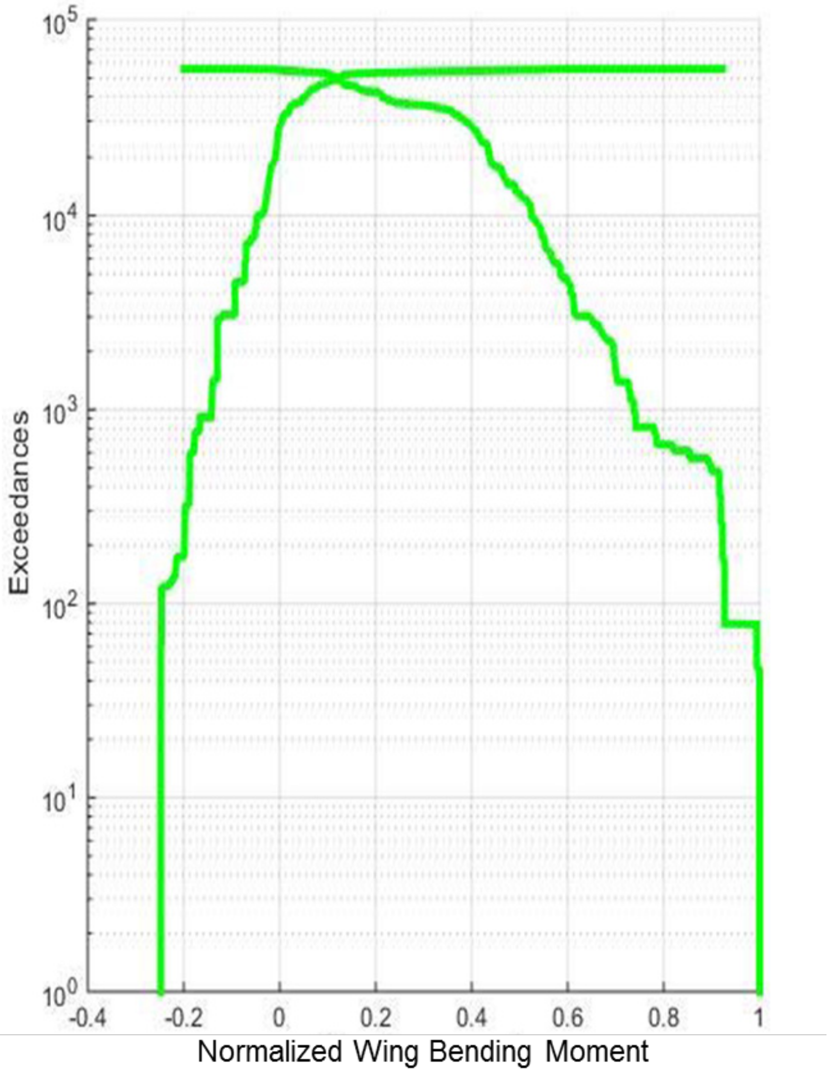


Figure C-2. Exceedance Curve for Modified Baseline MES.

In addition, the code for estimating flight parameters from loads was updated to require the flight parameters to remain within allowable limits. Flight parameters for the updated MES were generated, along with those for the marker bands.

According to initial calculations by Lockheed, the revised spectrum still seems to have relatively high levels of damage, though not as bad as before. One possible reason for this is Lockheed's assumption that the government supplied FEM is a coarse air-vehicle FEM. In reality, it is relatively fine for a typical air-vehicle FEM, so some amount of stress concentration may be present in the FEM results. Thus, additional stress concentration factors used in the control point stress analysis may be too large and the resulting life too short.

The final revision of the MES has a constant gross weight, as per a request from AFRL.

Appendix D – Implementation Details

P^{2}IAT Speed and Memory Improvements
Due to the time-marching scheme of the DBN method, each time step produces data. After refactoring, the total memory allocated at each step is nearly equal to the memory occupied by the actual data being produced (i.e. very little excess memory allocation is required). In addition, the Python data processing package Pandas is now used to organize and store all sample data. The package performs quite efficiently for large datasets, both in terms of memory and speed.

The required memory was minimized by using a HDF5 database. HDF5 is a high-performance database format for large datasets [28]. The data produced at each time step is dumped to an HDF5 database that is stored on disk. Routines for accessing and storing data were developed for both scalar and vector data. These routines are transparent to whether the code is run with standard RAM back-end or the HDF5 back-end. Most of the data used during the execution of the DBN only requires storing a single set of samples, and the current set is stored in RAM for fast access. When parts of the code require accessing previous sample sets, HDF5 access is relatively fast.

The memory footprint was further reduced by selectively choosing not to save certain computed values. For example, stress intensity factors are computed for both corner and through cracks for every flight. These can be easily reconstructed with the loads and crack size samples if necessary, and thus do not need to be stored. The network structure has a number of intermediate variables, so eliminating the storage of them reduced the memory footprint significantly (in one example from 14Gb to 700Mb).

Information-gain computations were implemented in a somewhat slow fashion for testing purposes. However, when doing inspection scheduling, many information-gain calculations must be performed, all with different assumed crack detection data. In order to speed up the process, the information gain calculations were integrated more deeply into the sampling routines.

The DBN implementation was already parallelized to take advantage of multi-core machines for the most expensive step, generating new samples which includes evaluating the BHM and crack growth models. During testing, an additional bottleneck was found when the code performed bounds checking on the samples. This procedure was also sped up by moving to a parallel implementation.

Restart capability was implemented in P^{2}IAT. Each run generates four files which store the model and data from the run. These files, along with the flight index at which to restart, can be specified in the input file to load the relevant samples as the initial state.

Since the main computational burden is the C-crack growth code, the relative speedup is better when the user specifies more flights per batch. That is, computing crack growth for 100 flights at a time is faster than one by one, and produces less data. However, one downside of this is that missing data can only be inserted on a per-batch basis. So, if single flights are missing, then the code must be run one flight at a time. In order to allow for missing flights in the data and do fast forecasting, separate number of flights-per-batch for data and forecast were implemented.

Thus, in the typical case, one would run with 1 flight-per-batch in data assimilation to estimate missing flights during a 100-flight interval, for example. And then move to 100 flights-per-batch for forecasting to speed up the forecast which may go out to 1000 flights.

In its current state, the code still has memory limits which are dictated by the fact that the C-crack growth code is a C-extension to python and compiled in 32bit python. Any 32bit application in Windows is subject to a 1-2Gb memory limit. Python overhead takes up much of this space. Thus, the current code is still somewhat limited in terms of the number of particles and flights-per-batch on Windows. Note that one can run the Python version of the crack growth engine in 64bit Windows, which removes the memory limit but is slower than the C-version. However, the code was successfully compiled in 64bit python on Linux and runs without memory limits. Thus, Linux is the preferred system for running the code. It seems to be quite difficult in general to compile Python C-extensions for 64bit Windows. If this problem can be solved in the future, the Windows limitation would be fixed.

Pre- and Post-Processing Routines

Since the generic DBN inputs are quite general, a preprocessing routine is required to convert standard flight recorder data into the required DBN input. The routine must:

1. Fill in missing flight recorder data within a flight
2. Add annotations to describe entire missing flights and forecasted flights
3. Add a "batch index" from specified number of flights per batch for data assimilation and forecast
4. Convert FRD to external loads using a BHM model. This is done in pre-processing to speed up execution time. The result is both mean and variance of bending and torque at station 3.
5. Rotate fixed input file so it can be specified by the user as 2 columns (the DBN code requires two rows).

The pre-processing routine facilitates modifying some parameters that are otherwise difficult to change in the generic DBN, like the number of forecasted flights.

The post-processing routine generates the IAT reports and maintenance reports in the desired format.

The P^{2}IAT framework also automatically generates sensitivity plots using the sample-based sensitivity metrics developed for the program. The plots are interactive when displayed in a browser.

Software Organization

Some software that was delivered is for model building (e.g. BHM, load spectrum generation, etc.). All of the other code is integrated with the generic DBN and consists of a number of packages and scripts. The code structure is shown in Figure D-1. The "crack_dbn_interface.py" script is the main interface between the generic DBN code and the crack growth and other routines written specifically for P^{2}IAT. The POD curves are just a set of simple, sigmoidal functions. The load forecasting script is a second set of code that interacts with the generic

DBN. In fact, it is a capability that should be included in the generic DBN codebase, but so far has not been fully integrated. In the meantime, it is included as an external script.

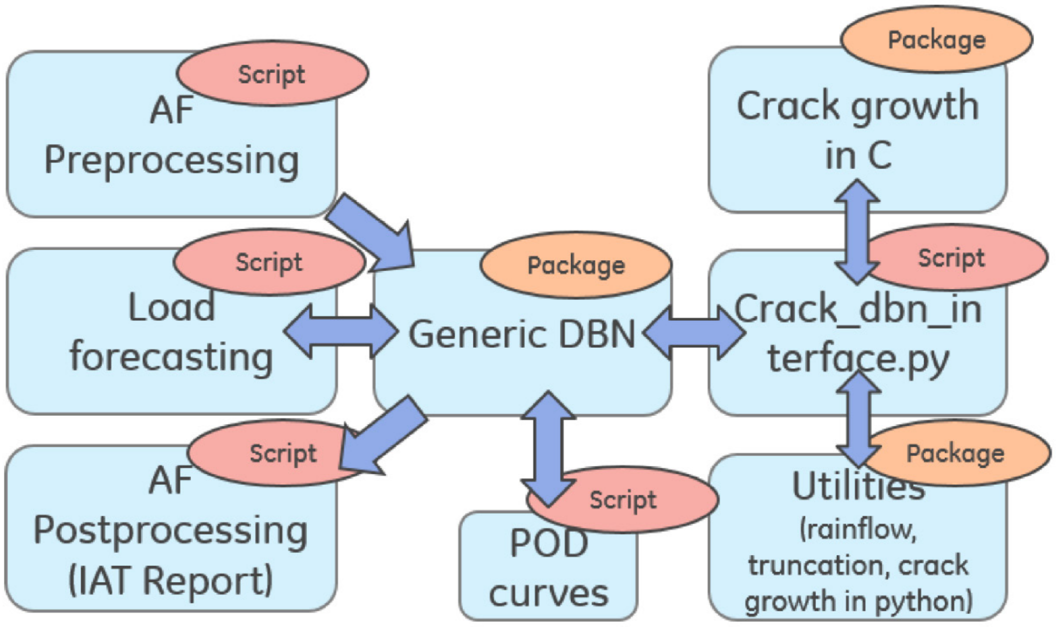


Figure D-1. Software Organization for Running DBN in P²IAT.

A standard data assimilation and/or forecasting run of the DBN requires a number of input files, which are described below.

- Main DBN run file. This file contains the network structure and points to all other files, including those defining distributions, models, data, and restart files. It also contains a number of settings for the methods. The file can also be set up to perform post-processing only for an existing finished run.

- Fixed_options.csv: all fixed parameters of the run, including number of flights to forecast, load forecast parameters, and any fixed variables (e.g. yield stress, plate thickness, and truncation).

- Inspection_options.csv: options for inspection scheduling, including turning on or off information gain, fixed interval, required interval, failure threshold, etc.

- Priors.csv: distributions for all uncertain inputs – material properties, load & geometry parameters, POD parameters, including extra files for any correlated inputs.

- Data_load_and_mission.csv: flight recorder data and mission data with a flight counter.

- Data_inspection.csv: inspection data (batch index and detection, either 0 or 1)

- Pout_xxx.mat, pout_xxx.skl: BHM or quadratic models for FRD to loads and loads to K_I

- MES: the baseline MES which is used for forecasting.
- IAT_aux_data.txt: auxiliary data for IAT reports, including dates/flight hours of previous reports, control point information, etc.
- Initial samples: if any prior distributions are specified by a collection of samples rather than a distribution.
- Restart files

Appendix E – Verification of the Python-Based Crack Growth Engine

The P^{2}IAT framework has shifted to center around a Python-based implementation of a Dynamic Bayesian Network method. This was chosen for improved speed and generality compared to the previous Matlab-based codes. The new implementation required re-writing code for load processing and growing cracks. The Python and Matlab implementations of the codes were both run in a deterministic mode with identical inputs. The processed loads and crack growth were compared and found to match, thus verifying the Python-based implementation.

The flight loads from one of the flight records provided by the Air Force were used for code verification. First, the load history was passed through extrema extraction and rainflow algorithms to extract the cycle content. The resulting spread of cycle amplitudes versus the stress ratio of the cycle for both the Matlab-based code and the Python-based code is shown in Figure E-1. The resulting cycle amplitude versus cycle number for both codes is shown in Figure E-2. The characterization of the load cycles in the flight is the same for both load processing algorithms verifying that the Python implementation is equivalent to the Matlab code.

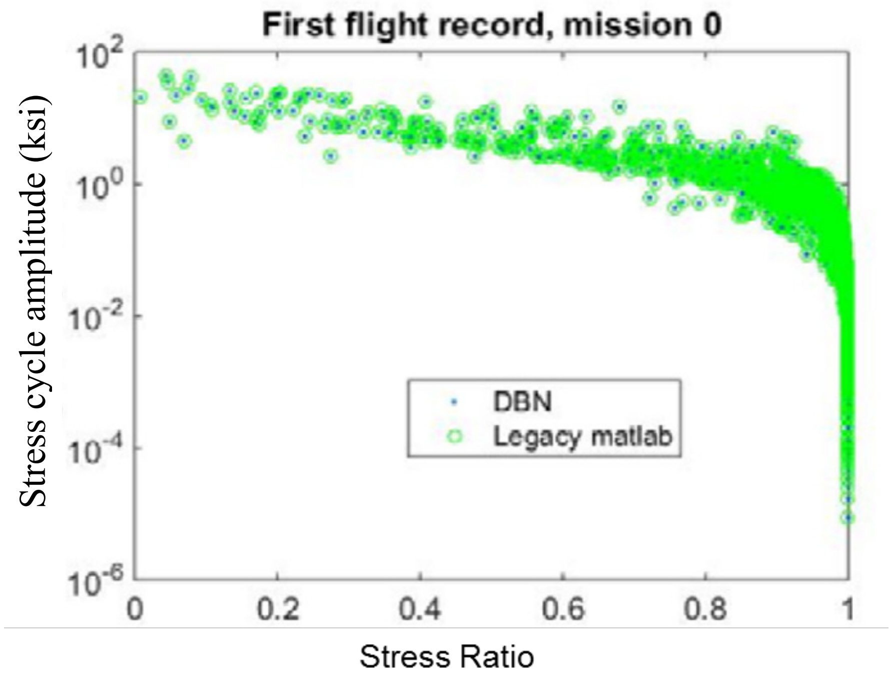


Figure E-1. Comparison of Cycle Amplitude vs. Stress Ratio for Python and Matlab Load Processing Algorithms. Vertical axis is on a log-scale.

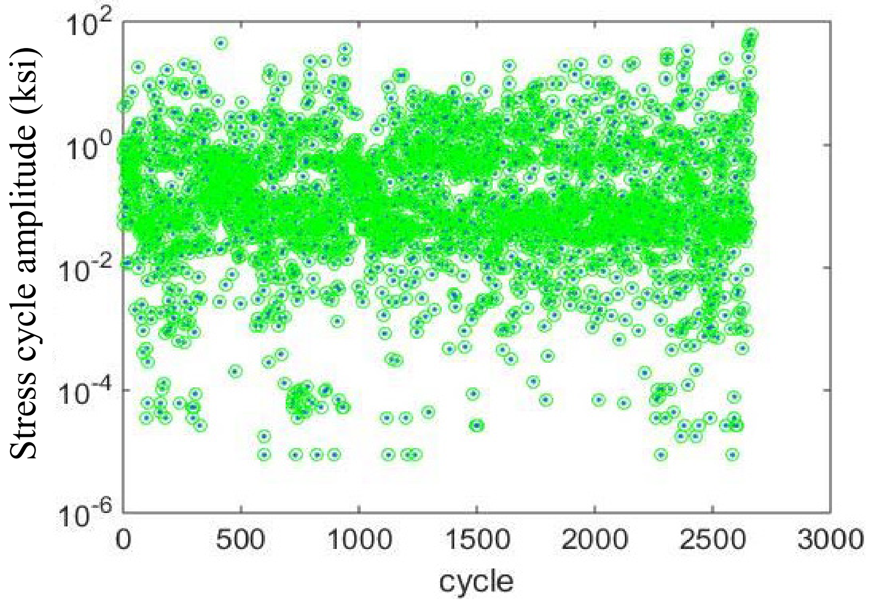


Figure E-2. Comparison of Cycle Amplitude by Cycle Number for Python and Matlab Load Processing Algorithms. Vertical axis is on a log-scale.

With the load processing verified, the cycles are passed through the two crack growth engines. The crack growth parameters, such as C_0, m, and γ, are set to the same nominal values in both codes. Figure E-3 presents the crack growth curves calculated by each code. The curves lay on top of each other. Figure E-4 shows that the two codes produce identical crack growth with differences on the order of machine precision (10^{-15}). The new Python-based implementation, which is faster and more general, has been verified to produce identical results to the legacy implementation. Although not shown explicitly here, the C-based implementation used in the final P^{2}IAT framework was verified against the Python version. Results were compared for crack growth with exactly the same inputs and the final crack length and plastic zone size was equivalent up to machine precision. Thus, all three implementations are consistent.

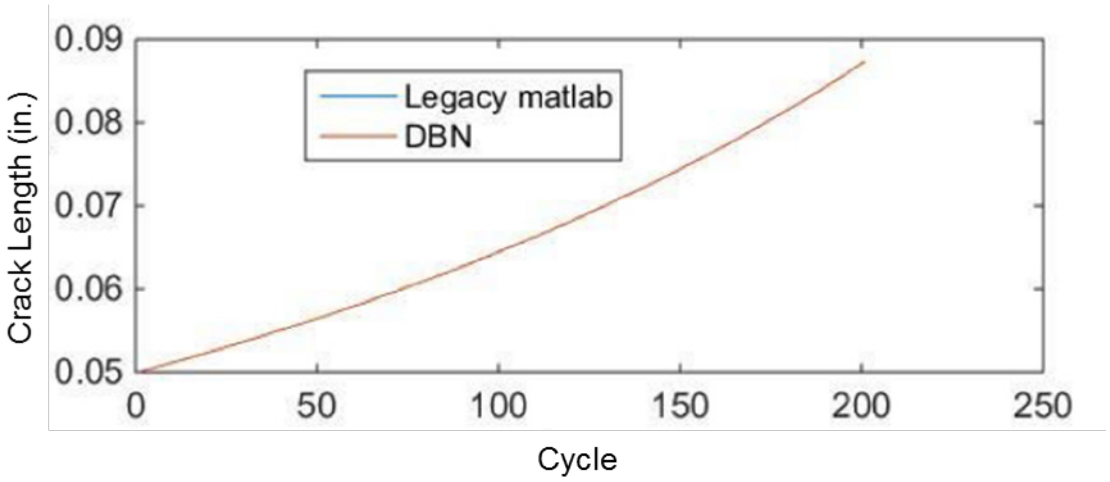


Figure E-3. Comparison of Crack Length Predictions for Python and Matlab Implementations.

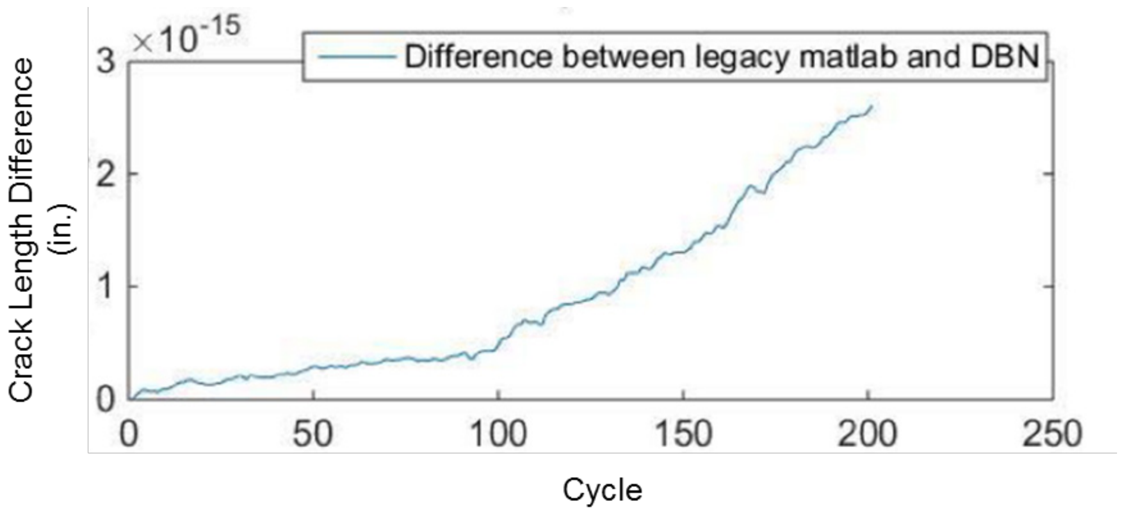


Figure E-4. Difference between Crack Length Predictions for Python and Matlab Implementations.

Appendix F – Inspection scheduling methods

Approach

There are numerous reasons for performing inspections of airframe structures. This section will focus on the following goals which have been discussed extensively with the Air Force.

1. To decide if the aircraft is safe to fly right now.
2. To predict (forecast) when the aircraft will be unsafe to fly.
3. To improve our understanding of damage progression and thus update models used to predict when the aircraft will be unsafe.

Contrasting these goals is the problem of cost – each inspection, depending on the control point of interest, requires bringing the aircraft out of service, into some facility, disassembly, and finally actual inspection. Some control points may be cheaper to inspect than others due to the amount of disassembly required to access them. In addition, multiple inspection methods can be employed on a single control point.

A-priori scheduling of inspection intervals is thus a difficult problem. One formulation of the problem is as a mixed-integer optimization problem. The variables to optimize are:

- Time of inspection (number of cycles/flights into the future, up to some time horizon)
- Locations to inspect
- Methods to use to inspect

The collection of all of these variables is termed the inspection schedule, see Figure F-1. The objective function for this optimization is less clear. It is some combination of risk and cost.

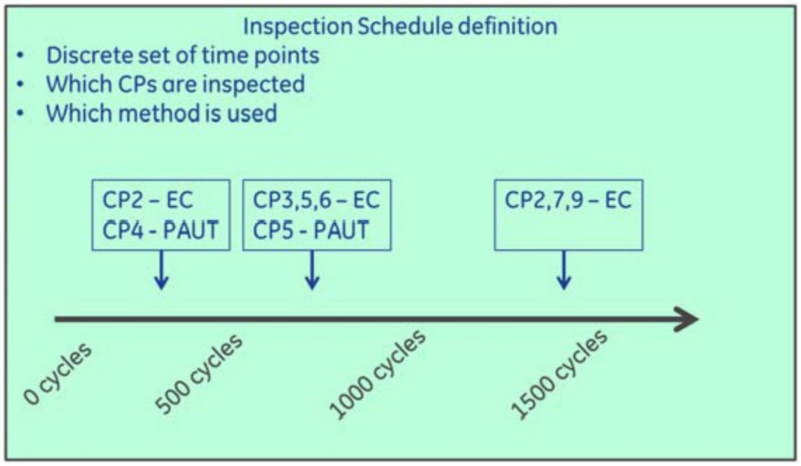


Figure F-1. Definition of an Inspection Schedule.

Some options are as follows:

1. Minimize cost for a given risk constraint.

2. Minimize risk for a given cost constraint.
3. Minimize both risk and cost.
    a. With a fixed tradeoff ratio.
    b. Return a family of solutions with various tradeoffs.
4. Find any solution that satisfies a risk constraint and a cost constraint.

Of course, each of these objectives will yield a different answer to what the "optimal" inspection schedule is. In order to formulate the problem fully, though, a number of other assumptions/definitions must be made which are detailed below. The discussion will be framed by considering how to evaluate the risk of a given inspection schedule. Once one can evaluate risk, one can move on to performing the optimization based on one of the above criteria and or constraints.

Detailed cost modeling will not be discussed here as it is assumed to be out of scope for P^{2}IAT. However, it is clear that it is desirable to limit the number of inspections. When performing the optimization of the inspection schedule, then, the number of inspections in a given period of time may be a good surrogate for cost.

Forecasted Usage

In order to design an inspection schedule and predict its outcomes, one must have some way to forecast the aircraft usage into the future. The forecasted usage will be probabilistic, meaning that there is an entire family of forecasted loads that obey some mean and variance criteria. This probabilistic usage will dictate a forecasted (probabilistic) crack length, i.e. an expected crack length distribution at every point in the future. The forecasted crack length can then be used to evaluate the expected risk of a given inspection schedule.

Thus, load forecasting must be done. Forecasted loadings will come in the form of a probabilistic load spectrum which has been developed for the Air Force Digital Twin program. The load spectrum was constructed by fitting a set of probabilistic models to real flight data as well as simulated data from the Stick-to-Stress V1.0 program. The load spectrum consists of 1,000 synthetic flight records, each of which includes loading information (the relevant loads for the experimental test are bending and torsion at the outer wing station). The flights are divided into five missions, some with more and some with less damage content. Initially, the load spectrum contains a particular proportion of flights of each mission type, i.e. a mission mix. The mission types and initial mission mix is shown in Table F-1.

Table F-1. Five Mission Types with Initial Mission Mix.

Easy	Baseline	Short hot	Moderate	Extreme
9.4%	18.8%	12.5%	28.1%	31.2%

To perform load forecasting, one can take a current mission mix and use it to randomly draw flights from the probabilistic load spectrum. During actual usage, it is likely that the mission mix will not be the same as the designed mission mix. Thus, the P^{2}IAT program includes a module that can learn the current mission mix when given data as to which missions were flown most recently. The most up to date estimate of the current mission mix is then used to draw forecasted flights. Note that, for this program, it is assumed that mission data is accurate and that a given mission type will, more or less, always correspond to the same type of loading (that is, it does not learn what each mission type means over time, but rather assumes that the labels in the probabilistic load spectrum are accurate). Such a feature is not difficult to implement, though it requires a large amount of flight data to be accurate.

The approach of forecasting usage by drawing from a load spectrum is different from the more common assumption of a fixed forecasted load, for example analyzing safety by assuming design limit loads or ultimate loads. This could be done, however it eliminates the uncertain nature of future loading and enforces a worst case, conservative loading.

A third option for load forecasting is the following. Consider a risk measure that uses the probability of failure using a fracture toughness criteria (rather than, for example, a critical crack size). The fracture toughness has as input the loads on the aircraft. One could imagine predicting crack size using the more realistic probabilistic load spectrum, while predicting failure (i.e. computing fracture toughness) using limit loads. In this case, the crack size evolution over time would be realistic (potentially non-conservative), while at any point in time the probability of failure would be computed using the realistic crack size with a conservative loading. Such a strategy would be justified under the assumption that while crack growth should be realistic, the pilot should be allowed to safely pull a maneuver that results in limit loads in the next flight (even if such an event is not expected).

Forecasting Inspections

Another issue that arises when evaluating an inspection schedule is the assumed results of an inspection. This includes:

1. The predicted inspection results in terms of a crack detected or not-detected.
2. Repair of a crack after any detection.

These assumptions influence the forecasted crack length and therefore the forecasted risk. When forecasting past an inspection, one should use the expected inspection results to update the model, just as would be done were the inspection to have actually occurred. One must therefore predict the results of the inspection. In P^{2}IAT, the inspection process is modelled with a probability of detection (POD) curve which is a function of crack length. For a given crack length, the POD curve returns the probability of detecting that crack. One way to predict inspection results is to use the forecasted crack length distribution and the POD curve. Using the computed probabilities of detection, one can draw (with a random number generator) the inspection result, either detected or not-detected. Each possible inspection result would have to be used to update the model (as would be done for a real inspection) and then propagate the crack growth forward in time.

In the case of the inspection resulting in detected, one must also decide if repair would occur. In real life it is likely that a repair would be made, which would reset the crack size distribution to the equivalent repair flaw size (ERFS) distribution and potentially change the physical parameters of the control point (e.g. diameter of the hole if it is reamed out). However, in the context of the experimental test in this program, it is likely that cracks will not be repaired.

The forecasting of inspection results can be a computationally demanding task. This is because a single sample of crack length results in a family of possible inspection results. And, each inspection result yields an entirely new crack length distribution when it is used to update the models. In order to simplify the computations, it is reasonable to collapse all of this branching uncertainty into just one distribution of crack length. Thus, for each sample of crack length, one would compute a single possible inspection result (of course drawn from the distribution) and then a single updated crack length (again drawn from the updated distribution). See Figure F-2 for a schematic of how this process works in P^{2}IAT. By using this process, the total number of samples needed in forecasting crack growth is kept at a reasonable number.

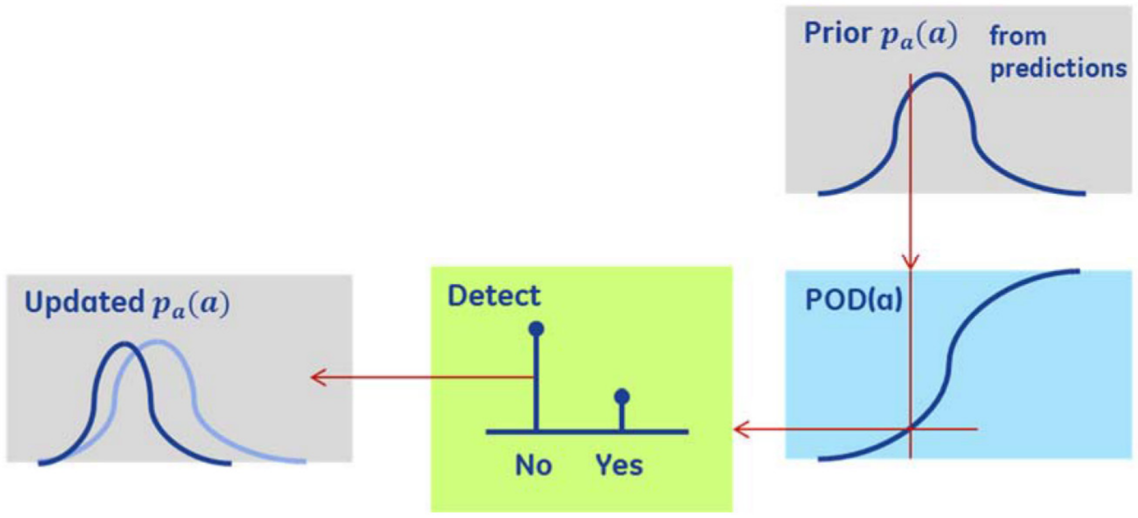


Figure F-2. Updating Process for Forecasted (Estimated) Inspections.

The approach described above is different than many standard approaches. Often, engineers will assume that every inspection returns not-detected and reset to lowest detectable crack. In contrast, the above approach would assume a family of possible inspection results and each result would be used to update the model. Note that the Bayesian updating process in P^{2}IAT will tend to shift down the crack length distribution when given not-detected data and increase it when given detected data.

Another approach to forecasting inspection results would be to treat every inspection as not-detected if this is deemed a reasonable assumption for scheduling purposes. This is certainly just a special case of the more general approach described above.

A note about the rogue flaw assumption is warranted here. The rogue flaw is considered to be a large crack that was in the structure from the beginning (outlier on the initial flaw size). The P^{2}IAT framework automatically considers a form of the rogue flaw assumption in all of its computations. When an inspection result is incorporated into the model, the framework does not automatically assume some pre-set crack size. Rather, it utilizes the POD curve to weight all of the possible crack sizes (i.e. the current crack length distribution) by how likely each one is to have been detected. Thus, rogue flaws are initially considered in the analysis and systematically determined to have lower and lower probability as inspections are performed. See Figure F-3 for a schematic of how inspection data are used to update the crack length distribution.

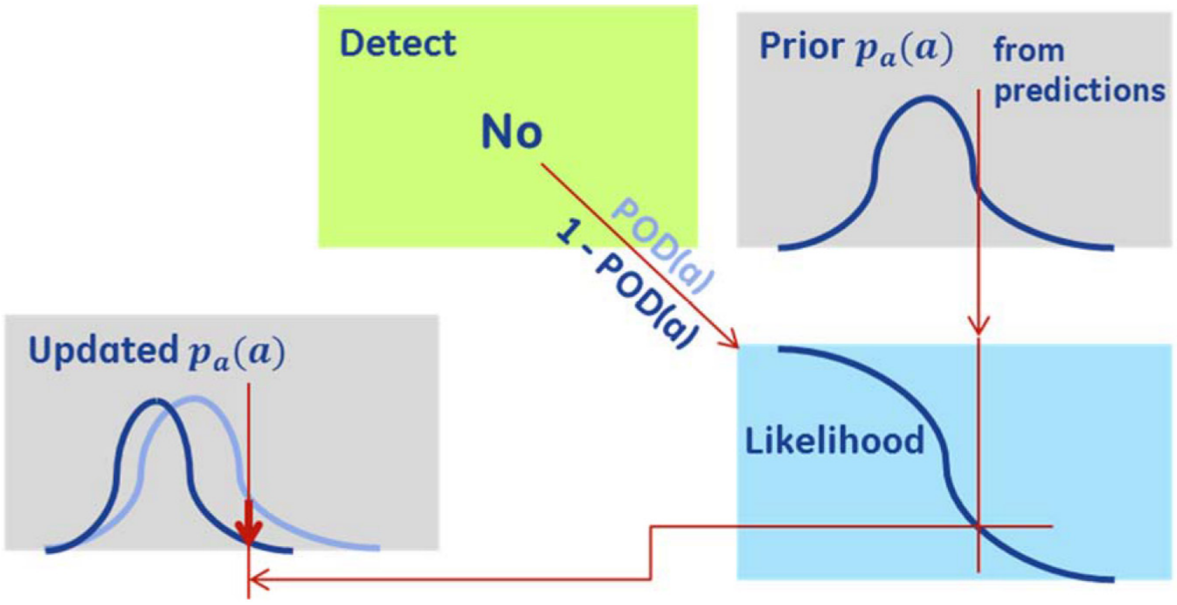


Figure F-3. Updating Process with Known Detection Data.

For example, suppose the inspection returned not-detected. And, suppose the current set of samples of crack length are [0.006, 0.008, 0.010, 0.020] inches. For each of these crack lengths, the POD is computed as, for example, [0.1, 0.2, 0.8, 0.99]. That is, a 0.020 in. crack is almost certainly detected, whereas a 0.006 in. crack only has a 10% chance of being detected. Each sample of crack size is then given the weights [0.9, 0.8, 0.2, 0.01]. The weighted samples constitute the new, updated crack length distribution. The code will take the weighted samples and draw a new set of samples that are equally-weighted but represent the same underlying distribution. Thus, the code does not preclude the possibility of the 0.020 in. crack existing (the potential rogue flaw), but rather says that it is not likely to exist (i.e., has a low probability of 0.01). As the crack growth is propagated and forecasted into the future, the rogue flaw is always propagated as well, albeit with a low probability of actually occurring.

As for computational implementation, the P^2IAT framework largely functions as a PF, which means that uncertainty is represented by a set of samples. Each sample of crack length is propagated forward in time, and the samples taken together (at any given time point) approximate the full distribution of crack length. Because the algorithm is sample-based, there is a fixed lower bound on the probabilities that can be represented. That is, if the user chooses to use 1,000 samples, then the smallest possible probability number that can be represented is 0.1% (i.e. one of the 1,000 samples). Thus, any tail probabilities less than this value are effectively truncated. With 1,000 samples, then, only rogue flaws with a likelihood of 0.1% or greater can be represented.

There are, however, ways to represent smaller probabilities in P^2IAT and more can be included as desired. In general, very small probabilities must be represented with a continuous probability distribution rather than samples. One way to do this is to fit a distribution to a set of samples (this requires specifying a form of the distribution, e.g. normal or log-normal). A second way is to compute samples of the probability directly rather than samples of crack length. This is currently done for failure probabilities so that probabilities of failure on the order of 10^{-7} can be represented. Finally, note that when the UKF/PF hybrid method is used, the crack length is represented by a normal (or lognormal) distribution for all P^2IAT computations, so the rogue flaw would always be treated correctly. However, in this case the accuracy of crack growth usually suffers due to the assumptions that are necessary for the algorithm.

Defining Risk
Obviously, the definition of risk is very important when evaluating the risk of an inspection schedule. When simulating cracks in metallic structures, risk usually refers to the risk of catastrophic failure due to a large crack which prevents the structure from bearing the applied loads. In material testing, a parameter called fracture toughness is usually measured which, for a particular loading condition and crack size, gives a criteria which determines whether or not the crack will grow in an unstable manner leading to failure. Given the variations in geometry and loading in the experiment as well as real life parts, one generally works with a distribution of fracture toughness which then results in a probability of failure. For the selection of the load to consider in the fracture toughness criteria, see the discussion above on forecasting usage. The load could be the expected/forecasted load or some fixed (e.g. limit) load.

Another possible criterion for failure is a critical crack size. The critical crack size could be determined by analyzing potential unstable crack growth under some fixed loading, e.g., small, typical, or limit loads. Alternatively, a critical crack size could be known based on geometry. For example, if a crack at a bolt hole grows toward another bolt hole, it may be reasonable to assume that once it reaches the second hole it will quickly grow to the point of failure.

While either of the above criteria are possible options in P^2IAT, the focus has been on the fracture toughness criteria. In this case, at every time point a set of samples of probability of failure are computed, one for each sample of the crack length. One must then summarize these probabilities of failure into a single risk metric. One example is to take the average probability of failure, another would be to take the value at the upper 75th or 95th percentile of the probability of failure. In the case of the average, constraining the risk below 10^{-7} would equate to requiring that the probability of failure is less than 10^{-7} with 50% probability.

An important consideration in the definition of risk is the inaccuracy in the models. Model inaccuracy is reflected in the distribution of crack length. Thus, if the models are very inaccurate, the crack length distribution will be very wide. A wide crack length distribution often leads to a significant probability of failure, since large cracks are possible. This probability of failure, though, is due more to the lack of knowledge in P^2IAT rather than an accurate prediction of failure. If the P^2IAT framework does not have much data, the predicted uncertainties will be large.

If one were to design inspection schedules based on a model that has little data, the result would be constant inspections (and model updating) until the model predicts narrower crack length distributions (Figure F-3). This strategy, however, may be unacceptable to the users who may need to move forward with little data. One way to get more reasonable inspection schedules with an ill-informed model is to trust the mean of the model. This could be done, for example, by artificially reducing the variance of the initial flaw size (or of the crack length distribution), the variance in the applied loads (e.g. forecast usage by just repeating a single flight), or the variance in other sources of uncertainty (e.g. material properties, load conversion, etc). Note that it is quite simple to modify these distributions in the P^{2}IAT framework.

Risk Constraint Approach: Information Gain per Dollar

Consider a number of possible inspection schedules for a single control point and a risk threshold constraint. Now consider forecasting from some initial crack size distribution into the future. The risk threshold defines a point in time at which an inspection must be performed simply due to the risk level. The benefits of performing the inspection are:

1. Deciding if the aircraft can safely fly now.
2. Updating the model.
    a. Reduced model uncertainty usually leads to less risk.
    b. Potentially the crack is growing slower than expected; more time before next inspection.
    c. Potentially the crack is growing faster than expected; less time before next inspection.

If the aircraft is unsafe, then usually a repair will be performed and the crack length distribution will be re-set to the initial flaw size distribution (smaller). Otherwise, no crack will be detected and the crack length distribution will also be smaller.

Now consider a point in time between the current time and the risk threshold defined inspection (t_{risk}). A natural question is the following: does there exist a point in time *before* the risk threshold inspection (t^*) at which it would be better to perform an inspection compared to waiting t_{risk}? The only way that inspecting at t^* would be better is if the inspection provided more information for model updating than the inspection at t_{risk}. This would push the next-defined t_{risk} further into the future, resulting in fewer inspections overall. Unless the crack ends up growing faster than expected, in which case it is better to have an earlier detection anyway.

The above argument naturally leads to the information gain criteria for performing inspections. Probabilistic methods can be used to compute the expected information gain (IG) at any point in time. The IG is a function of the predicted crack length distribution and the uncertainty in the inspection. This is represented by the POD curve for yes/no crack detections. The resulting inspection schedule is determined by maximizing the information gain from the current time to t_{risk}. With a cost model of the inspections, one can also compute the information gain per dollar and use that as the criteria to optimize within the interval.

When used as the inspection criteria, the information gain fuses the model forecasts with the inspection methodology. This results in some interesting border cases; see Figure F-4 for a schematic.

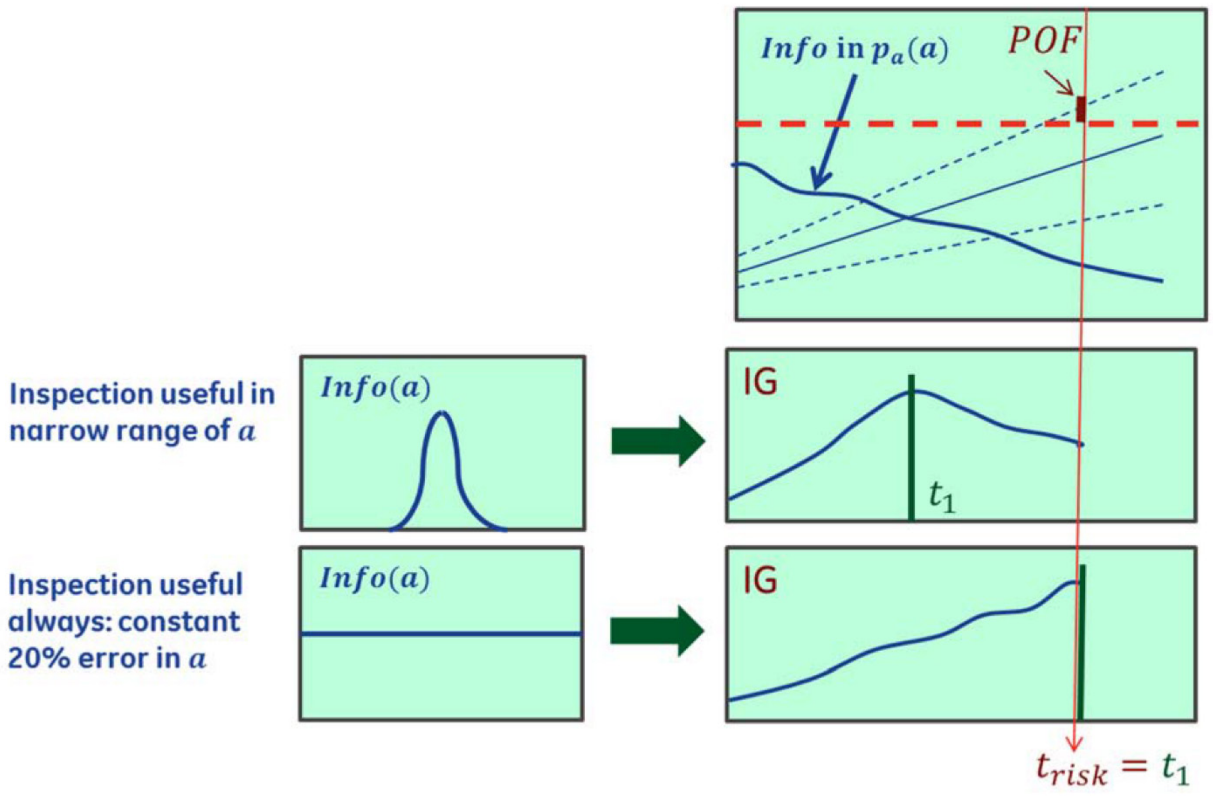

$Info$ in $p_a(a)$
POF
Inspection useful in narrow range of a
$Info(a)$
IG
t_1
Inspection useful always: constant 20% error in a
$Info(a)$
IG
$t_{risk} = t_1$

Figure F-4. Information Gain from Predictions to Updates with Two Types of Inspections.

Suppose the inspection always has the same error (suppose one can always measure crack length to within 20%). Then the measured crack length distribution always has the same uncertainty (width), so the information in the measurement is always the same. Over time, forecasts will tend to have increasing levels of uncertainty due to unknown operational conditions. One can say that the forecasts have a decreasing amount of information over time. Thus, the information gain always increases with time. In this case, the optimal time for inspection is always at t_{risk} (furthest possible time into the future).

Suppose the inspection method can only perform well at a narrow range of crack lengths. Then the information in the measurement will peak at a point in time when the crack length reaches that narrow range. Suppose that the forecasting has relatively little growth in uncertainty, so the information in the forecast is constant. Then, there will be a particular point in time between the current time and t_{risk} at which the information gain is maximum – when the forecasted crack length is in the narrow range where the inspection method performs well.

In reality, the level of accuracy in the inspection is described by the width of the POD curve (for detections) and the uncertainty in the forecasts is determined largely by previous updating and assumptions. The situation may be somewhere between the extremes described above. Calculating the IG over time allows one to determine, for the particular situation at hand, where the optimal inspection should be placed.

The process for placing one inspection can be repeated into the future to place all inspections. To do this, one must make an assumption about what occurs after an inspection (e.g. repair if a crack is found). Generating an inspection schedule in this way certainly respects the risk threshold and is likely to result in an overall low cost, though it is not guaranteed to do so.

Minimizing Risk for a Given Cost

Consider the problem of minimizing risk for a given cost at a single control point. Suppose that a cost constraint can be translated into a fixed number of inspections. Then, the problem is to apportion out these inspections over a given time horizon. Most inspections would result in a reduced mean crack length, so the highest risk times are the flights right before the inspections. Thus, the overall maximum risk level can be computed as the maximum of the risk levels at each inspection and the risk level at the end of the simulation time. This guards against the case where inspections are all placed early and the risk grows unacceptably high before the end of the simulation.

If one wants to minimize the risk, one solution is to have risk equally spread among all inspections and the final time. This amounts to performing inspections whenever the risk reaches a threshold, but the threshold is now determined by the number of allowed inspections. For practical reasons, of course, it is much easier to specify the risk threshold and then determine where to place inspections. The minimum risk value could be computed, for example, by stepping though a series of risk thresholds and finding the maximum risk threshold that results in the desired number of inspections.

Combining Inspections at Multiple Control Points

It is of interest to not only reduce the total number of inspections but also to group inspections into a limited number of time points at which the aircraft will be disassembled to perform the inspections. While one could formulate the problem as optimizing over all possible combinations of inspections at each control point, such a formulation could make the optimization algorithm quite slow as there are so many possible combinations. A simpler approach, which is still realistic, is to identify inspection times for each control point separately and then aggregate them into groups. This greatly simplifies and speeds up the process of optimizing the inspection schedule for individual control points.

Determining rules for aggregating inspections is a difficult task that must make a few assumptions. Suppose the individual scheduling for control point A suggests an inspection should be performed after 500 cycles, while for control point B an inspection at 550 cycles is suggested. The questions that must be answered are:

1. Should these inspections be performed at the same time?
2. If so, should they be performed at 500 cycles, or 550 cycles, or some time in between?

In order to answer these questions, one must go back to the goals of the inspection. If the goal of immediate safety is paramount, then the conservative approach would be to inspect both A and B at 500 cycles and again at 550 cycles. However, this may be quite expensive while the level of risk may not decrease much by doing all four inspections. Note that considering all possible combinations and evaluating a risk level for each would devolve back to the large computational expense problem discussed above. It is of interest to devise a set of rules that results in acceptable risk while lowering cost.

One such set of rules could be the following:

1. Go to the first inspection indicated (due to risk).
2. Look ahead for nearest indicated inspections. For each one, consider adding it to the inspection plan at this time if:
    a. POD at the current time is decently high (maybe >10%).
    b. If inspection comes back negative, next scheduled inspection would be far out in the future.
3. Once an inspection plan has been finalized for this time point, assume all of the relevant data are collected, update models, and continue forecasting until the next indicated inspection at a CP. Continue from 1.

The strategy described above is a greedy one. A greedy strategy attempts to optimize a quantity locally in a series of steps with the goal of achieving a globally optimal solution. In this case, the greedy approach means that one does not consider every possible future scenario when deciding where to place an inspection. Rather, one looks only a little ways into the future (i.e. only consider what might happen at the next inspection or two) to determine what to do at the current time. In contrast, a globally optimal approach would consider all possible sets of inspections from the current time until the aircraft is taken out of service. Of course, a global approach may be able to find a better solution in terms of lifetime maintenance cost of an aircraft, but it will be very computationally challenging given the large number of possible future scenarios. The greedy approach will not necessarily yield the most optimal inspection schedule. It will, however, return a decent schedule that is likely to be better than a fixed-interval schedule.

Input from AFRL *Face-To-Face Meeting*
During the face-to-face meeting in Dayton, OH on May 11-12, 2016, the criteria for inspection scheduling were discussed at length. We agreed with AFRL on a number of details about how P^2IAT would do inspection scheduling. First, we will not try to optimize the inspection schedule. That is, some criteria for how to place inspections will be set in place. Then P^2IAT predictions will be used to sequentially place inspections forward in time. No attempt will be made to go back and modify the inspection placement in order to have a more desirable schedule.

The risk criteria to use is the expected single-flight probability of failure (SFPOF) greater than 10^{-7}. Overflying this criteria is possible, but should be rare, and the expected SFPOF should never exceed 10^{-5}. Inspections need to be performed before risk reaches this level. In addition, inspections can only be performed at particular intervals. Field-level inspections should be performed every 200 flight hours and depot-level inspections should be performed every 1800 flight hours or flights. Depot-level inspections also usually involve disassembly of the wing structure.

Inspections will be simulated during forecasting and inspection scheduling. That is, once an inspection criteria is met, the P^2IAT framework will simulate the result of the inspection using the current guess of the crack size and the inspection method capability (POD curve). The framework will further simulate the expected reduction in uncertainty due to updating the model with the results of the inspection. This should reduce the probability of failure prediction.

The framework will also have the option to simulate repair by re-setting the crack size distribution and potentially changing other parameters.

GE will use the information gain criteria to schedule inspections as well. The inspection time of the information gain criteria must not exceed that of the risk criteria (i.e. it must respect the risk threshold). The information gain criteria has been described in the previous sections.

From now on GE will use the following additional settings:

1. SFPOF will be calculated using a limit load in the residual strength calculation. Crack growth forecasts will still be done with the MES (using the most recent updated mission mix).
2. In forecasting, each sample should have a different (random) flight.

Results
The inspection scheduling methodology presented above was implemented in the P^{2}IAT framework. Four cases for the inspection schedule generation for the demonstration location were compared.

1. Baseline forecast: no inspections or repairs
2. PDM: 200-cycle inspection/repair interval
3. Risk based: establish an inspection whenever the log-mean SFPOF exceeds 10^{-7}. That is, for a set of samples of SFPOF, the criteria is $\exp\left(mean(\log(SFPOF))\right) > 10^{-7}$. The log-transformation ensures reasonable scaling when the distribution of SFPOF spans many orders of magnitude. The time point at which the risk based inspection occurs is called t_{risk}.
4. Information-gain based:
    a. Forecast crack growth until t_{risk}.
    b. Compute the information gained (IG) at each point from the previous inspection up to t_{risk}. This is done by predicting the inspection result using the forecasted crack length and computing the expected information gained in the crack length distribution if the predicted inspection data were used to update the models.
    c. Set the inspection at the flight with the highest IG between the previous inspection and t_{risk}.

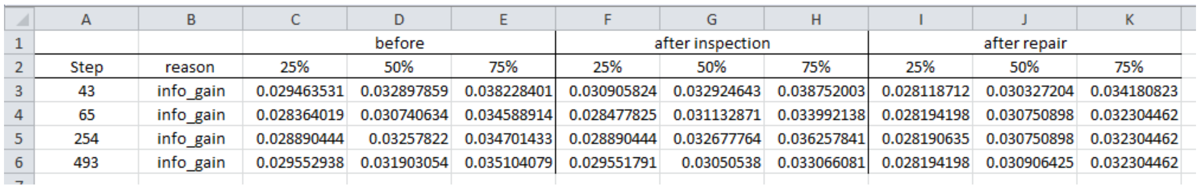
In all of the above scenarios, the following settings were used:

- Repair is assumed to occur if a crack is detected; no repair occurs if a crack is not detected. Note that since the detection is estimated probabilistically (a binomial distribution of detect/no-detect), the repair is also probabilistic. Upon repair, the crack size is re-set to the initial crack size. This follows the assumption that a repaired bolt hole has the same equivalent initial flaw size distribution as the original, as-manufactured hole did. No assumption is made about the possibility of a poor quality repair resulting in potentially larger initial flaw sizes, or about the possibility that the stress near the hole changes due to the repair. The framework, however, does allow one to explore such scenarios.

- The initial crack size is set intentionally large to have reasonable values of SFPOF. The crack size is log-normal with $\mu = -3.5, \sigma = 0.2$. This gives a mean crack size of about 0.03 inches. This is basically a detectable crack for the assumed POD curve ($a_{90/95} =$ 0.01368 inches).
- Fracture toughness is set to a normal distribution with $\mu = 32$ ksi and $\sigma = 1$. This is a reduced standard deviation from previous examples because a large σ gives unreasonably high SFPOF. In any case some more work needs to be done to establish the correct distribution for fracture toughness for each control point specifically.
- SFPOF is calculated using a limit load with a bending moment of 1.2×10^6 lb-in.
- The forecast was done using the version 3 MES (with constant gross weight). The forecast was done out to 1000 flights using the baseline mission mix.
- A simplified version of the stress intensity factor model was used for this analysis. The model takes into account the hole radius, thickness, crack length, and wing bending moment at station 3.The risk and information-gain based scheduling approaches are allowed to schedule inspections after any flight, not just at the PDM intervals of every 200 flights.
- 50 samples were generated per time step. This small number was used for demonstration purposes.

An example of the resulting inspection schedule is shown in Table F-. At each inspection, the crack length distribution is reported before the inspection, after the inspection, and after the repair (if the crack is repaired). The distribution is characterized by the 20th, 50th, and 75th percentiles, though more can easily be computed. The reason for the inspection can either be the information gain criteria, the risk criteria, or both. The step indicates the number of flights from the beginning of the simulation at which the inspection occurs.

Table F-2. Inspection Schedule for an Example Run.



		before			after inspection			after repair		
Step	reason	25%	50%	75%	25%	50%	75%	25%	50%	75%
43	info_gain	0.029463531	0.032897859	0.038228401	0.030905824	0.032924643	0.038752003	0.028118712	0.030327204	0.034180823
65	info_gain	0.028364019	0.030740634	0.034588914	0.028477825	0.031132871	0.033992138	0.028194198	0.030750898	0.032304462
254	info_gain	0.028890444	0.03257822	0.034701433	0.028890444	0.032677764	0.036257841	0.028190635	0.030750898	0.032304462
493	info_gain	0.029552938	0.031903054	0.035104079	0.029551791	0.03050538	0.033066081	0.028194198	0.030906425	0.032304462

Crack growth curves for the four P^2IAT runs are shown in Figure F-5 to Figure F-8. The baseline scenario with no inspections shown in Figure F-5 shows the crack to grow quite large by 400 cycles. Scheduling inspections every 200 flights as in Figure F-6 keeps the mean crack size relatively low, though the upper tail of the crack length distribution at the first inspection is large. Scheduling inspections based on attaining a specific risk level results in just two inspections and also keeps the crack length relatively small as shown in Figure F-7. The information gain based approach shown in Figure F-8 has one inspection right at approximately flight 8 and does seem to have a relatively large tail of large crack sizes for the remainder of the 1000 flight interval.

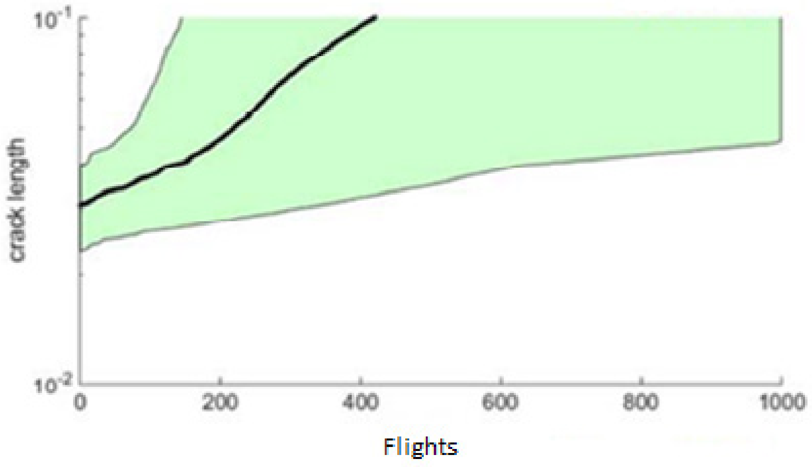


Figure F-5. Crack Growth and Uncertainty with no Inspections.

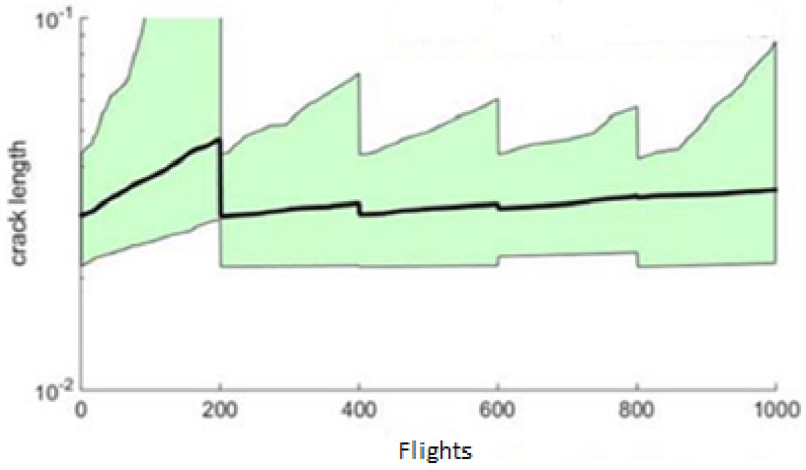


Figure F-6. Crack Growth and Uncertainty with Inspections Every 200 Flights.

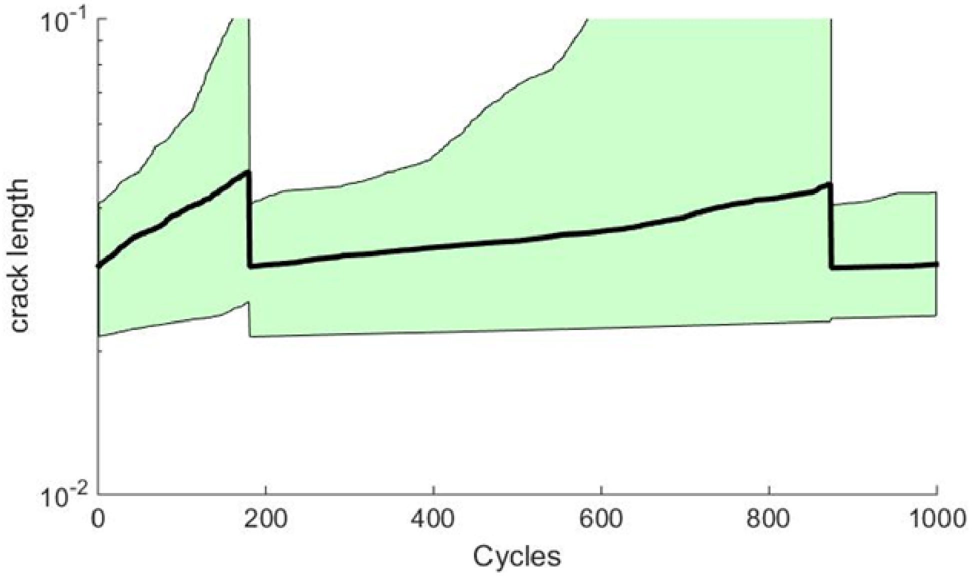


Figure F-7. Crack Growth and Uncertainty for Inspections Based on Risk Threshold.

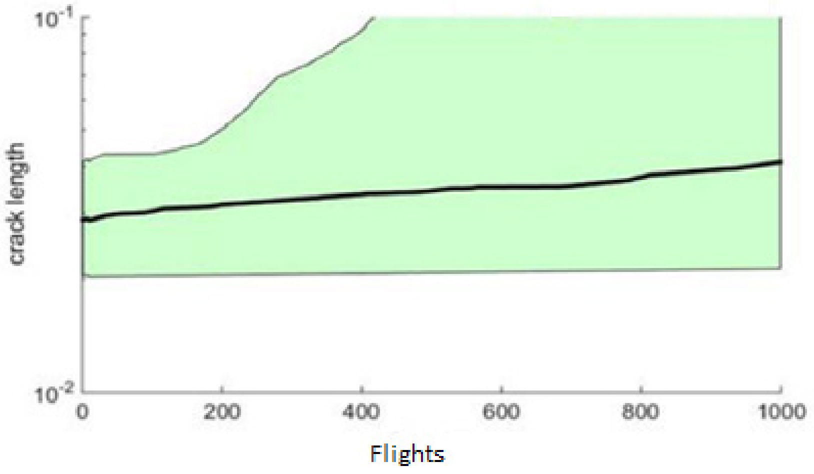


Figure F-8. Crack Growth and Uncertainty for Inspections Based on Information Gain.

More insight can be gained by examining the SFPOF curves, shown in Figure F-9 to Figure F-12. The yellow horizontal bars represent the region where $10^{-7} < SFPOF < 10^{-5}$ and the red bars represent $SFPOF > 10^{-5}$. The baseline scenario with no inspections in Figure F-9 shows that the risk threshold is reached around 200 flights and failure is very likely to occur by 300 flights. The plot for inspections every 200 flights in Figure F-10 shows that the initial inspection is performed at a relatively high risk. After that inspection, though, the model is updated and the crack growth slows substantially. Subsequent inspections do little to update the model as the crack grows slowly.

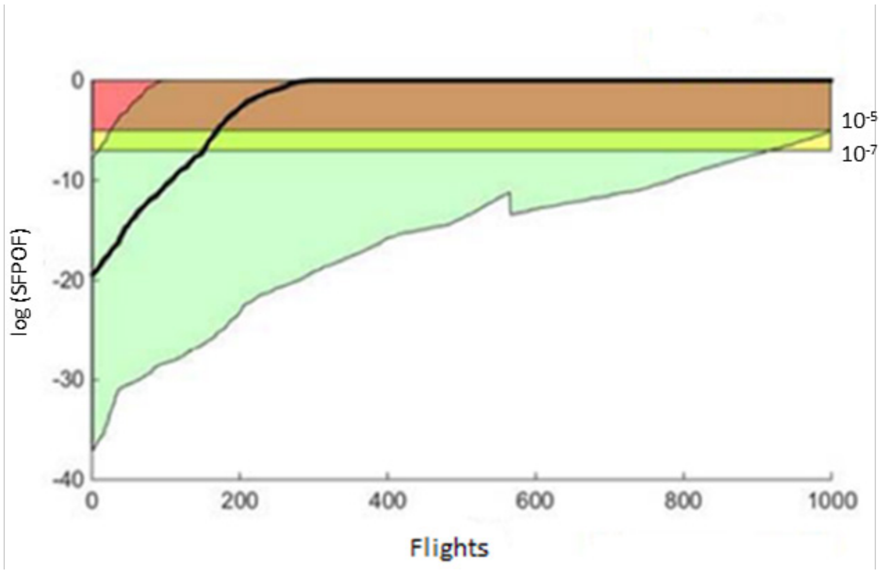


Figure F-9. SFPOF and Uncertainty for Baseline Scenario with no Inspections.

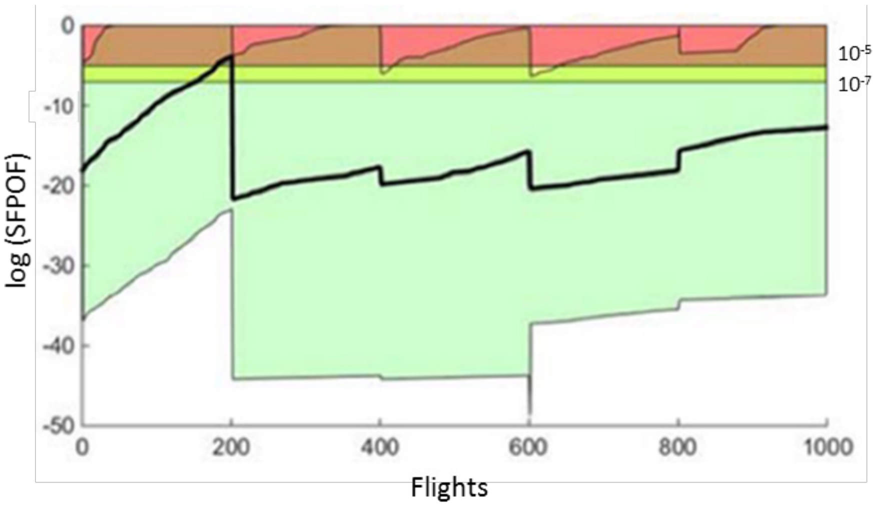


Figure F-10. SFPOF and Uncertainty with Inspections Every 200 Flights.

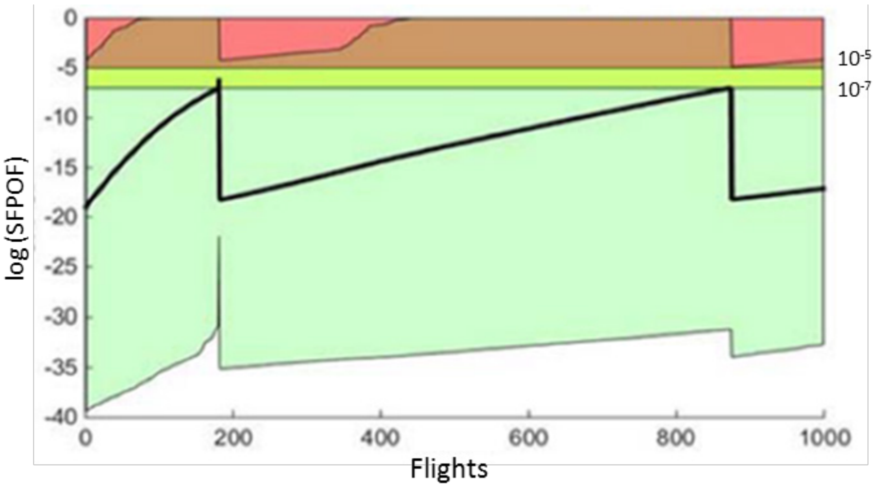


Figure F-11. SFPOF and Uncertainty for Inspections when SFPOF Exceeds 10^{-7}.

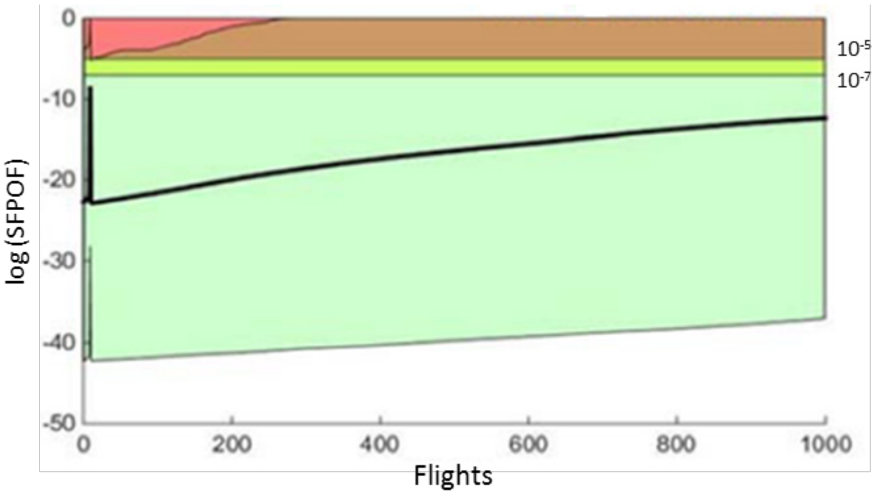


Figure F-12. SFPOF and Uncertainty for Inspections when Maximum Information is Gained.

The risk based scheduling approach in Figure F-11 results in the first inspection occurring slightly before the 200 flight point. After this, the model is updated and the crack growth rate slows so that the next inspection occurs beyond 800 flights. The risk based scheduling reduces the number of required inspections from five when inspections are scheduled every 200 flights to two.

The information-gain based scheduling in Figure F-12 has the first inspection occurring right near the beginning of the 1000 flight block. This is because the crack length distribution starts out in a range where any crack has a reasonable probability of being detected. With continued loading, P^2IAT is so certain there is a large, easily detected crack present that only an inspection that does not find a crack would affect the model. Thus, the initial inspection is done very early so that the model is updated significantly. This results in relatively little crack growth for the remainder of the 1000 flights. Note that in this particular example, the crack tends to grow slower than expected at time zero. In general this may be the opposite. In fact, the crack grows so little that the mean SFPOF never reaches 10^{-7} by the end of the time period. Thus, the information gain based inspection criterion results in just one inspection. The plot also shows that while the tail of the crack length distribution does go relatively high as seen in Figure F-8, the mean SFPOF does not. Thus the tail of the crack length distribution is relatively thin.

The error bars on the plots show the uncertainty in the SFPOF due to the crack length distribution. However, note that the overall probability of failure is a single number which encompasses all of the uncertainty in the system, and this number is the mean (black line) on the plot. Thus, one must be careful in interpreting the upper end of the error bars as “it might have a large SFPOF.” Rather, the correct interpretation is “The mean SFPOF is x. Thus there is some possibility of failure, however small.”

In summary, the four inspection scheduling approaches result in the following:

1. Baseline: no inspections, likely failure at 300 cycles.
2. 200 flight interval: 5 inspections, max SFPOF = 1.7×10^{-4}.
3. Risk based: 2 inspections, max SFPOF = 9.0×10^{-7}.
4. Information gain based: 1 inspection, max SFPOF = 4.2×10^{-9}.

The 200 flight interval and risk-based methods result in repairing the crack but the crack growth rate after the repair is not as slow as it is in the information gain based method. This is because the 200 flight interval and risk-based methods perform inspections near the failure point when the crack is very large. Large cracks are outside of the range where the inspection method is informative. The cracks are always detected, so the inspection does not help update the model much. The information gain method, in contrast, performs the inspection while the crack is more in the middle of the POD curve, resulting in a more informative inspection, a more updated model, slower future crack growth, and ultimately no need for further inspections in the 1000 flight hour window.

Note that this example is a particular case where the inspection intervals, crack growth rate, initial crack size, and POD curve all result in superior performance of the information gain method. There are other cases where information gain is equivalent to the other methods and some cases in which the other methods can be better.

- If the inspection is not useful until right before SFPOF reaches 10^{-7}, then information gain will reduce to the risk based method.
- If there is very large uncertainty in crack length and loads, the information gain method may schedule very early inspections which are not really needed.
- If the user has input low uncertainty in the parameters but incorrect values for crack length (i.e. the model is confident about the crack length but it is wrong), the inspection will be in the complete wrong place, potentially far too late. An interval method would avoid this by insisting on inspections at regular intervals. For perspective, though, note that this problem can always occur when one trusts models to assess damage and delay inspections.

Another detail to note about the inspection scheduling approach is the importance of the time horizon. The method forecasts out to t_{risk} and then backtracks to find the time to inspect. In the example shown above, after the first inspection at flight 8, the forecast hits the time horizon of 1000 flights before it reaches the next t_{risk}. Thus, no further inspections are scheduled.

Consider what would have happened if the time horizon was extended to 2000 flights. The forecast would have continued out to the next t_{risk} at, say, 1500 flights. At this point, the algorithm would again backtrack to set the inspection. This could put the second inspection anywhere from flight 9 to flight 1500. Perhaps the inspection would occur at, say 400 flights in. Thus, with a larger time horizon, it is possible to have a different inspection schedule for the first 1000 flights! It is not valid to divide the aircraft lifetime up into small pieces and schedule inspections for each piece and then concatenate them. The algorithm would give a different answer if it was working with the entire timeline altogether. Considering larger timelines will, however, result in much longer computational run-times. Some balance must be made in this regard depending on the computational resources available. The downside of considering small intervals can be alleviated using a heuristic though. If one sees the information gain continually decreasing as the aircraft ages, it is an indication that an inspection should have been performed earlier. At that point it is best to perform the inspection as soon as possible to gain the most information from it.

Computationally, the simulations took about 30 minutes to run on a single processor. The simulations were for just 1 control point and had only 50 samples to describe the uncertainty in the system. The full solution file with distributions for every variable for all 1000 flights was over 3 Gb. More recent updates to the framework have reduced much of the overhead and computational time.

Appendix G – Verification of SFPOF method

The method used for calculating the single flight probability of failure (SFPOF) in P^{2}IAT is essentially the same as the method described by Halbert [5]. The method takes advantage of the Bayesian updating formula. In Bayesian updating, a model is updating with data by computing the conditional probability of the model given the data:

$$P(Model|Data) = \frac{P(Data|Model)P(Data)}{P(Model)} \tag{G-1}$$

The updated distributions correspond to the term on the left. Thus, Bayesian updating is essentially the same as conditioning on the data.

The definition of SFPOF is the probability of failure in the next flight *given* that the system has not yet failed. Thus, SFPOF is a conditional probability measure. In P^{2}IAT, the probability of failure can be computed at any time using the critical stress intensity factor. By supplying the data that the system has not failed, the failure calculation is conditioned and is therefore computing SFPOF directly.

If one were not to include the not-failed data, the framework would predict a myopic, instantaneous probability of failure. That is, the computed failure probability would not take into account the fact that the system recently had some likelihood of failure. This can occur, for example, if a harsh mission is followed by a benign mission. The failure probability in the benign mission is small, but SFPOF would be larger because of the possibility of failure during the previous harsh mission.

The calculation of SFPOF by P^{2}IAT can be verified using a simplified problem that has an analytical solution. As a test problem, consider an asset which has applied load and some internal resistance to that load. When the resistance is less than the applied load, failure occurs. The resistance and load are defined as known probabilistic functions of time shown in Figure G-1. The resistance and load are normally distributed at each time point. The sinusoidal load simulates loading cycles or periods of harsh and benign operation. Note that each cycle is considered to be statistically independent.

The analytic method first computes the probability of failure at time n from the initial time until time n. This cumulative failure probability is $F_N(n)$. The SFPOF at any time can then be computed as

$$SFPOF(n) = \frac{F_N(n)-F_N(n-1)}{1-F_N(n-1)} \tag{G-2}$$

The analytic solution for $F_N(n)$ is

$$F(n) = \int_0^\infty \{1 - \prod_{k=1}^{n} F_{S(k)}[y(k)]\} f_{R(1)}[y(1)]dy(1) \tag{G-3}$$

Samples of resistance are taken to compute the outer integral with Monte Carlo. The inner cumulative probabilities are computed analytically assuming the load S is normally distributed.

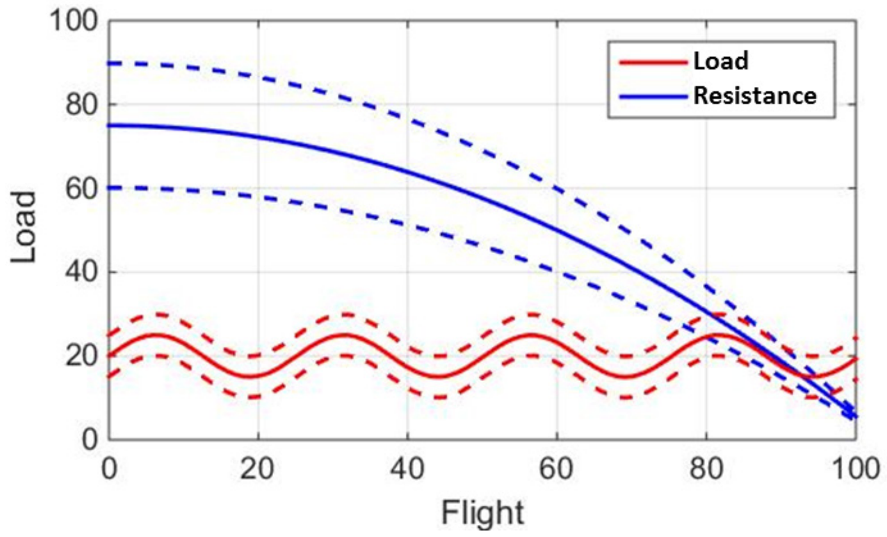


Figure G-1. Example for Verifying SFPOF Calculation.

Note that an important aspect of the P^{2}IAT approach is that the failure criteria consists of a known distribution for the fracture toughness. For metals, one can often assume this to be a normal distribution. The analytic distribution for fracture toughness is used in conjunction with samples of crack length and loading to give samples of SFPOF at each step. That is, the Bayesian network generates a set of samples $SFPOF_i = \Phi\left(\frac{K_I^i - \mu_{K_{Ic}}}{\sigma_{K_{Ic}}}\right)$ where K_I^i is the maximum stress intensity factor for the i^{th} particle and $\mu_{K_{Ic}}, \sigma_{K_{Ic}}$ describe the fracture toughness distribution. The final, total SFPOF is simply the arithmetic mean of the individual particle SFPOF samples. An alternative approach would be to take samples of the fracture toughness itself as part of the DBN process. Doing so would lead to a strong dependence on the number of samples used in the DBN. Computing accurate probabilities of failure on the order of 10^{-7} would require around 10^8 samples which in many cases is too large for the available computational resources. By keeping the fracture toughness distribution analytic, much smaller SFPOF values can be accurately resolved with fewer samples.

The analytic and DBN (P^{2}IAT) methods are compared to a naïve Monte-Carlo approach where both the load and resistance are calculated. All three methods used 10,000 samples. The results are shown in Figure G-2. The DBN method shows good agreement with the analytic method even down to machine precision. The Monte Carlo method shows good agreement but cannot compute any value of SFPOF less then around 8×10^{-4} due to the sample size.

In order to see the effect of adding the fail=True data to the DBN, the true solutions for SFPOF and the instantaneous, myopic probability of failure are compared. Figure G-3 shows (in a linear scale) the difference between the true SFPOF and the instantaneous probability of failure. In this case, the SFPOF is smaller because a system which survived a previously harsh load condition is more likely to survive later benign conditions. The DBN solution, which is meant to compute the true SFPOF by incorporating fail=True data, follows the true SFPOF curve.

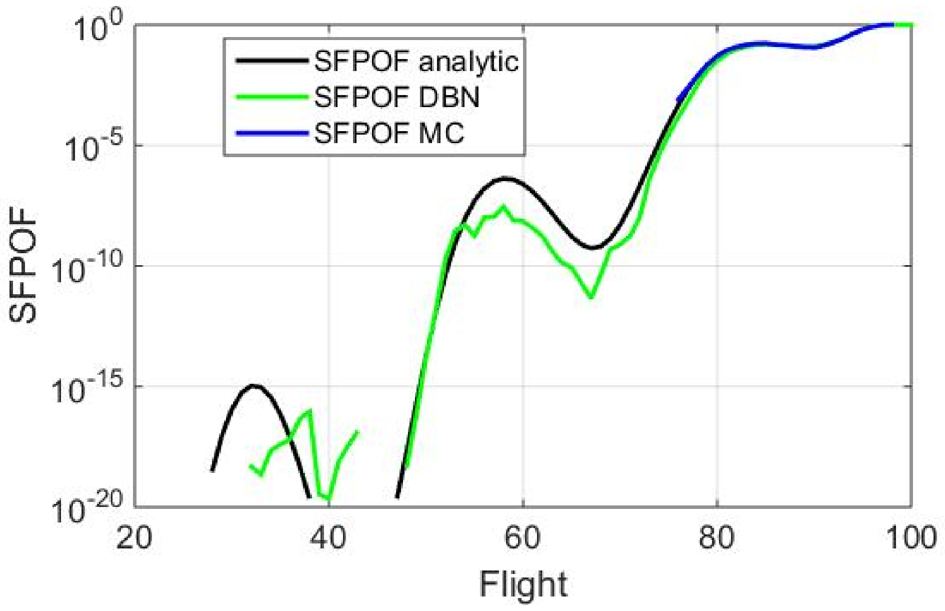


Figure G-2. Comparison of three methods for computing SFPOF.

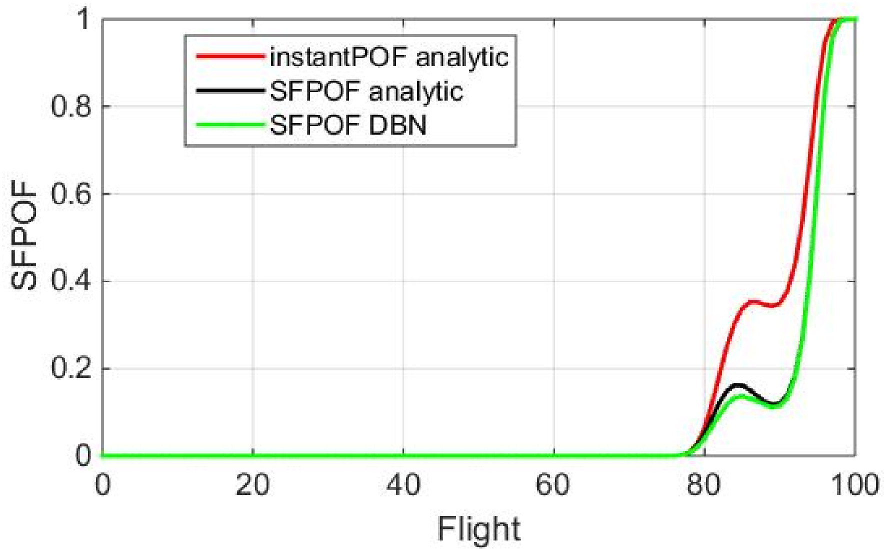


Figure G-3. Comparison of DBN and analytic methods for SFPOF calculation.

The DBN method in P^{2}IAT has been validated to compute the correct values for SFPOF on an analytic test problem. The DBN method is much more general than the analytic method. The DBN method works when:

- the loads are correlated within a flight and flight-to-flight, both of which are true for many real applications,
- the resistance is correlated in time, which is always the case in crack growth applications,
- or the models are updated using other data sources, e.g. inspection data.

All of these cases are important ones where the analytic method cannot apply.

LIST OF SYMBOLS, ABBREVIATIONS, AND ACRONYMS

ACRONYM	**DESCRIPTION**
ANN	Artificial Neural Network
API	Application Program Interface
ASTM	American Society for Testing and Materials
DBN	Dynamic Bayesian Network
DOE	Design of Experiments
DTDH	Damage Tolerant Design Handbook
EIFS	Equivalent Initial Flaw Size
FE	Finite Element
FEM	Finite Element Method
FRD	Flight Recorder Data
GE	General Electric
GEBHM	GE's Bayesian Hybrid Modeling
IC	Initial Conditions
IDACE	Intelligent Design and Analysis of Computer Experience
IG	Information Gained
KL	Kullback-Leibler
MCMC	Markov Chain Monte Carlo
MES	Master Event Sequence
NDI	Non-Destructive Inspection
P2IAT	Prognostic and Probabilistic Individual Aircraft Tracking
PF	Particle Filter
POD	Probability of Detection
SFPOF	Single Flight Probability of Failure
SIR	Sampling Importance Resampling
SIS	Sequential Importance Sampling
UKF	Unscented Kalman Filter
WBR3	Wing Bending Right Side at Station 3
WTR3	Wing Torque Right Side at Station 3